NASA/TM—2003-212001/REV1

SCPS–TP, TCP, and Rate-Based Protocol Evaluation

Diepchi T. Tran and Frances J. Lawas-Grodek
Glenn Research Center, Cleveland, Ohio

Robert P. Dimond
Verizon FNS, Inc., Brook Park, Ohio

William D. Ivancic
Glenn Research Center, Cleveland, Ohio

May 2005

The NASA STI Program Office . . . in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA Scientific and Technical Information (STI) Program Office plays a key part in helping NASA maintain this important role.

The NASA STI Program Office is operated by Langley Research Center, the Lead Center for NASA's scientific and technical information. The NASA STI Program Office provides access to the NASA STI Database, the largest collection of aeronautical and space science STI in the world. The Program Office is also NASA's institutional mechanism for disseminating the results of its research and development activities. These results are published by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA's counterpart of peer-reviewed formal professional papers but has less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or cosponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services that complement the STI Program Office's diverse offerings include creating custom thesauri, building customized databases, organizing and publishing research results . . . even providing videos.

For more information about the NASA STI Program Office, see the following:

- Access the NASA STI Program Home Page at ***http://www.sti.nasa.gov***

- E-mail your question via the Internet to help@sti.nasa.gov

- Fax your question to the NASA Access Help Desk at 301–621–0134

- Telephone the NASA Access Help Desk at 301–621–0390

- Write to:
  NASA Access Help Desk
  NASA Center for AeroSpace Information
  7121 Standard Drive
  Hanover, MD 21076

NASA/TM—2003-212001/REV1

SCPS–TP, TCP, and Rate-Based Protocol Evaluation

Diepchi T. Tran and Frances J. Lawas-Grodek
Glenn Research Center, Cleveland, Ohio

Robert P. Dimond
Verizon FNS, Inc., Brook Park, Ohio

William D. Ivancic
Glenn Research Center, Cleveland, Ohio

National Aeronautics and
Space Administration

Glenn Research Center

May 2005

Acknowledgments

The authors would like to thank the following people who either offered technical support on protocol problems, offered their expertise in conducting tests and analyzing results, or provided hardware resources needed by the research team: Keith Scott, Pat Feighery, Eric Travis, Charlie Younghusband, Adrian Hooke, Brian Adamson, Mark Allman, and Jim Griner.

Document Change History

This revised report, numbered as NASA/TM—2003-212001/REV1, May 2005, supersedes the previous version, NASA/TM—2003-212001, September 2003.

Page 9, equation (1) has been changed to read

$$\text{Bandwidth} = 0.93 \times \frac{\text{MSS}}{\text{RTT} \times \sqrt{p}} \qquad (1)$$

Page 9, last paragraph: Last sentence has been deleted.

Pages 27 to 45, appendixes B to I: Remove ± symbols and insert parentheses around values. A footnote has been added to read "Numbers in parentheses are standard deviations."

Report Documentation Page, Supplementary Notes: Organization code has been changed to RCN.

This report contains preliminary findings, subject to revision as analysis proceeds.

Available from

NASA Center for Aerospace Information
7121 Standard Drive
Hanover, MD 21076

National Technical Information Service
5285 Port Royal Road
Springfield, VA 22100

Available electronically at http://gltrs.grc.nasa.gov

Contents

SCPS–TP, TCP, and Rate-Based Protocol Evaluation

Diepchi T. Tran and Frances J. Lawas-Grodek
National Aeronautics and Space Administration
Glenn Research Center
Cleveland, Ohio 44135

Robert P. Dimond
RS Information Systems, Inc.
Brook Park, Ohio 44142

William D. Ivancic
National Aeronautics and Space Administration
Glenn Research Center
Cleveland, Ohio 44135

Summary

There have been numerous discussions relative to the role of the Space Communications Protocol Standards Transport Protocol (SCPS–TP) in the ever-evolving Transmission Control Protocol (TCP) family, particularly with respect to other known TCP features that accommodate high-bandwidth and long-delay networks (e.g., large windows and selective acknowledgments (SACKs)). To gain a better understanding of the performance of the SCPS–TP relative to other TCP variants and to determine the maturity of the various options for use on higher rate space links, the NASA Space Communications and Data Systems (SCDS) Office requested that Glenn Research Center perform a comprehensive set of tests. The goals were to validate the operation of the reference implementation of the SCPS–TP relative to its controlling CCSDS (Consultative Committee for Space Data Systems) specification and to perform a comprehensive comparison of SCPS–TP options and other TCP options.

We studied the effect of delay and bit error rate (BER) on the performance of congestion-based and rate-based protocols in emulated space links under uncongested conditions and with limited congestion. The results correlate well with previous testing of TCP options, including the SCPS–TP:

(1) The single-stream and multiple-stream test results clearly illustrate that the Vegas enhancements to TCP (that are implemented by the SCPS–TP) provide performance improvements over a TCP–SACK implementation tested in high-bandwidth-delay-product environments. Since performance considerations are subjective, the operational value of these performance increases can best be assessed by the users of specific applications hosted within those environments.

(2) Very small transactions such as command and control will probably see little difference in performance for any variant of TCP. In extremely error-prone environments with high round trip time (RTT) latencies, use of a rate-based TCP variant such as that provided by SCPS–TP is advisable, assuming that the network implementation is properly engineered. Users are cautioned against using rate-based protocols on networks that contend for resources according to allocation policies.

(3) The tested SCPS–TP implementation was the protocol reference implementation, which exists as a user application rather than as a more generalized protocol service provider. For some deployments, an in-kernel protocol service provider may prove more desirable than the deployment of application level protocol service providers. The tradeoff is between the more efficient use of resources and possibly higher

performance for in-kernel services versus the ease of maintainability and deployment of application-level implementations. Commercial in-kernel and application-level implementations of SCPS–TP exist but have not been tested as part of this effort.

(4) Even with equal performance, the SCPS–TP version of a rate-based protocol may be more desirable to implement than other rate-based protocols (such as the Multicast Dissemination Protocol, (MDP)) because the SCPS–TP is capable of requiring only sending-side modification. This feature of the SCPS–TP eliminates many of the risks associated with requiring universal deployment of new software.

(5) Existing TCP implementations (drawn from a variety of communities and including the SCPS–TP) appear to satisfy all currently known space mission needs; however, the space mission community should maintain an awareness of current and future TCP research that is being performed by other communities.

Introduction

In the late 1980s and throughout the 1990s, the Internet has rapidly developed allowing vast improvements in communication and networking. These technologies utilize packet-based communications rather than circuit-based communications. The Consultative Committee for Space Data Systems (CCSDS) foresaw the need to take advantage of this new Internet technology and developed the Space Communications Protocol Standards (SCPS) to address some specific issues related to space systems. Thus, the TCP/IP (Transmission Control Protocol/Internet protocol) suite was investigated and modifications to the networking, security, and transport protocols were specified. These specifications are known as the SCPS Network Protocol, Security Protocol, Transport Protocol, and File Protocol (SCPS–NP, SCPS–SP, SCPS–TP, and SCPS–FP, respectively).

There have been numerous debates regarding the actual improvements that SCPS may provide over the ever-evolving TCP/IP suite. In addition, much of the initial SCPS testing and demonstrations often did not provide what many consider to be a valid comparison relative to TCP, as known improvements to TCP for high-bandwidth, long-delay networks were often not implemented (e.g., large windows and selective acknowledgments (SACKs)) (ref. 1). Other testing was performed over simulated links where SCPS would provide little advantage because of the very low bandwidth (ref. 2). Some well-documented and thorough testing has been performed at lower rates. These results correlate well with our test results (refs. 3 and 4).

To better understand the actual improvements, if any, that SCPS could provide relative to TCP and to determine the maturity of the various protocols for higher rate links, the NASA Space and Data Communications Systems (SCDS) Office requested that Glenn Research Center perform a comprehensive set of tests. This report documents the tests performed to validate the operation of SCPS–TP (relative to the CCSDS specification) and to provide a comprehensive comparison of SCPS–TP options and other TCP-based protocols. This testing was only performed for SCPS–TP: Neither the SCPS–SP nor the SCPS–NP was implemented or tested.

Appendix A lists the acronyms used throughout this report as an aid to the reader.

Protocol Overview

TCP

TCP is a reliable transport protocol that uses a sliding-window-based congestion-control algorithm proposed by Van Jacobson (ref. 5). In particular, TCP congestion-control methods include *slow-start* (ref. 6), congestion-avoidance, fast-retransmit, and fast-recovery algorithms (ref. 7). The *slow-start*

algorithm is activated (triggered) at the beginning of a transfer or after a retransmission timer timeout (RTO) and occurs until either the congestion window (*cwnd*) reaches the *slow-start* threshold (*ssthresh*) or packet loss occurs. During the *slow-start* phase, if the receiver buffer size is large enough, the number of segments injected into the network is doubled every round trip time (RTT). When the *cwnd* exceeds the *ssthresh*, the congestion-avoidance algorithm is used to lower the sending rate by increasing the *cwnd* at most by one segment per RTT. This is the additive-increase algorithm of TCP and is done to probe for additional network capacity. Upon the arrival of three duplicated acknowledgments (ACKs) at the sender, the fast-retransmit algorithm is activated to retransmit the indicated segment without waiting for the RTO to expire. Duplicate ACKs may occur when a packet is lost or reordered by the network, and yet three additional packets arrive at the receiver. After the retransmission of the lost segment, the fast-recovery method is used to adjust the *cwnd*. As a result, *ssthresh* is set to half the value of *cwnd*, and then the *cwnd* is divided by two and three segments are added. At this point, for each duplicate ACK that is received, the *cwnd* is increased by one segment until the ACK of the retransmission arrives. After that, *cwnd* is set to *ssthresh* and the additive-increase algorithm is activated until *cwnd* is equal to the advertised window or until loss is detected, indicating possible congestion.

Since the above fast-retransmit method can only fix one lost segment per RTT, the subsequent lost segments within that RTT usually have to wait for the RTO to expire before being resent. In addition, an aggressive sender can retransmit segments that may have been received. The combination of the SACK option (ref. 8) and fast-retransmit and/or fast-recovery algorithms can be used to solve these problems. With the SACK and timestamp options, the receiver informs the sender about segments that have been received so that the sender can recover multiple lost segments within an RTT.

For most variants of TCP currently used in practice, including TCP-Reno and TCP–SACK, the sending rate is cut in half each time a loss occurs. The sending rate is then gradually increased until another loss occurs. This process is known as additive increase, multiplicative decrease, and is repeated until all of the data has been transmitted. This is one of the reasons TCP has difficulty operating efficiently[1] over long-delay, error-prone networks.

SCPS

Van Jacobson (VJ).—SCPS–VJ has the same congestion-control mechanism as described in the previous section for TCP except that SCPS–VJ uses the selective negative acknowledgment (SNACK) (ref. 9) option which is adapted from NACK, negative acknowledgment (ref. 10). Assuming packet reordering has not occurred, SNACK identifies specific lost segments that need to be retransmitted and can inform the sender of multiple lost packets at a later time in a bit-efficient manner. In addition, since SNACK does not depend on the fast-retransmit algorithm to resend the lost segments, the sender does not need to wait for the three duplicated ACKs that may never arrive in an environment with a high bit error rate (BER) or in an asymmetric network with an extremely slow return link.

Vegas.—Under this section, options specified in "TCP Vegas: New Techniques for Congestion Avoidance" (ref. 11) are presented, and then the modifications to this code is described as incorporated under SCPS-Vegas.

Whereas the VJ method saturates the network and uses lost segments as an indication of congestion, TCP-Vegas tries to avoid congestion in a network before it experiences losses. The Vegas algorithm tries to predict the congestion by monitoring the variations of the throughput and adjusts the *cwnd* based upon this throughput measurement. As a result, the sending rate in Vegas can be reduced before losses occur.

[1]TCP will reliably transfer all data over a long-delay, error-prone network. However, for current TCP deployments, a single TCP flow will not fully utilize the available bandwidth under such conditions.

The following are descriptions of the retransmission, congestion-avoidance, and modified *slow-start* mechanisms as used in TCP-Vegas:

(1) Retransmission.—First, Vegas records the system clock each time a segment is sent and its corresponding ACK arrives. Then, Vegas uses these times to calculate the RTT of the segment and a timeout period for that segment based upon the RTT. Second, when a duplicated ACK is received, Vegas retransmits the segment if its timeout period has expired without waiting for three duplicated ACKs to arrive. Third, when a nonduplicated ACK of the first or second segment after the retransmission arrives, Vegas checks the timer of this segment again to see if it is expired. If it has, Vegas retransmits the segment. This retransmission helps to fix any other segments that may have been lost before the retransmission without waiting for duplicated ACK. In addition, in the case of multiple losses, the *cwnd* in Vegas is reduced only when the dropped segment was sent after the last decrease of *cwnd*.

(2) Congestion avoidance.—TCP-Vegas adjusts the *cwnd* based on the difference between the expected throughput and the actual throughput (difference = expected – actual). Also, two thresholds, *alpha* and *beta* (*alpha* < *beta*), are defined as indicators of too little and too much extra data in the network respectively. If the difference is less than *alpha*, which indicates that the network capacity is large enough to achieve the expected throughput, the *cwnd* is increased linearly during the next RTT. If the difference is greater than *beta*, which implies a sign of congestion occurring in the network, the *cwnd* should be decreased linearly during the next RTT. If the difference is between *alpha* and *beta*, the *cwnd* remains unchanged.

(3) Modified *slow-start*.—Vegas allows the *cwnd* to double its value only every other RTT to detect and avoid congestion during the *slow-start* phase. Vegas leaves the *slow start* and enters linear mode when the actual throughput falls below the expected throughput by the equivalent of one router buffer or when loss has occurred.

In the implementation of SCPS Reference Implementation (SCPS–RI) version 1.1.62, the MITRE Corporation made several modifications to the above original *slow-start* and congestion-avoidance mechanisms of “TCP Vegas: New Techniques for Congestion Avoidance” (ref. 11). Reference 11 indicates that the Vegas *cwnd* can increase exponentially only every other RTT during the *slow-start* phase. The congestion-control window in the *slow-start* phase of SCPS-Vegas was changed in the 1.1.62 implementation to double upon the arrival of the first ACK in every RTT, ensuring that the *cwnd* would grow exponentially every RTT (equivalent to the *slow-start* process in SCPS–VJ and TCP).

In the congestion-avoidance phase, the *cwnd* in the original Vegas algorithm is decreased by one packet when the difference between the expected throughput and actual throughput is greater than *beta*. In SCPS-Vegas, the *cwnd* is reduced by half the amount of packets that the difference is over than *beta*.

SNACK and delayed ACKs were enabled while testing this SCPS-Vegas option.

There is also an SCPS-Vegas corruption option, which has the same retransmission, congestion-avoidance, and *slow-start* algorithms as in the SCPS-Vegas congestion option except for the protocol’s reaction to packet loss. The *cwnd* under SCPS-Vegas congestion is reduced by half assuming packet loss is due to link congestion, while the *cwnd* in SCPS-Vegas corruption assumes the loss was not due to congestion and leaves *cwnd* unchanged.

Pure rate control.—The SCPS-pure rate option does not activate the congestion-control algorithm. The sending rate depends on the values of (1) the rate option defined by the user and (2) the receiver buffer size. Like the TCP and SCPS–VJ tests, the acknowledgments for this test are delayed ACKs. The SNACK option, as described previously, was also enabled in the pure rate control tests.

There is an additional option of SCPS-pure rate that uses strictly delayed ACKs. Here, the ACKs are sent back every time interval as defined in the delayed ACK timer instead of being sent back every packet or every other packet. Using the delayed ACK timer as a trigger to send back an ACK can be beneficial in a long-delay environment where a longer period of time can pass before a second packet arrives.

MDPv2

The original Multicast Dissemination Protocol (MDP) (ref. 12) was developed between 1995 and 1997 as the underlying protocol for the image multicaster (ref. 13), a reliable multicast application developed in 1993 and also used on the multicast backbone (MBONE) for delivery of compressed image files and some bulk data content to multicast receivers. The protocol is based on the User Datagram Protocol (UDP), providing high throughput by avoiding congestion-control mechanisms in favor of a user-selectable rate. MDP achieves its reliability through the use of NACKs and can scale to provide efficiency in large multicast groups through NACK suppression (to minimize receiver message implosion) and the aggregation of control messages. In addition, the MDP method of operation adapts well to asymmetric links.

In 1997, Multicast Dissemination Protocol version 2 (MDPv2) (ref. 14) was developed and extended MDP capabilities to include a parity-based repair mechanism, an emission-control mode where clients refrain from message transmission, congestion-control algorithm, and tunable parameters, allowing MDP to adapt to a myriad of networking environments.

Glenn's testing utilized MDPv2, which was compiled from source code version 1.9a4, but modified with increased receive buffers. The application was used in a unicast network environment in a pure rate mode without congestion control or forward error corrections. The rate for the application was set at its optimum, which was determined to be 40 Mbps.

MFTP

First submitted to the Internet Engineering Task Force (IETF) in 1998 by Starburst Communications, Inc., the multicast file transmission protocol (MFTP) (Robertson, K., et al.: StarBurst Multicast File Transfer Protocol (MFTP) Specification. Internet Engineering Task Force, Apr. 1998, work in progress) consists of two protocol components: the multicast control protocol, and the multicast delivery protocol. Multicast control protocol is an administrative protocol allowing the server to dynamically control the joining and leaving of its multicast groups. The multicast delivery protocol handles the reliable transmission of data to the registered clients.

Much like its MDPv2 counterpart, MFTP utilizes a rate-controlled UDP/IP stream to transport data combined with a NACK scheme to provide TCP reliability while bypassing TCP congestion-control cost. Unlike MDPv2, MFTP does not provide any parity-based repair scheme or optional congestion control.

Glenn initially tested the MFTP application in support of shuttle mission STS–99 (ref. 15) to provide bulk data transfers for the German Aerospace Agency, DLR. Glenn testing utilized MFTP version 3.05 with the rate set at the application maximum of 15 Mbps.

Test Bed Configuration, Procedures, and Options

Configuration for All Tests

The Glenn test bed environment (fig. 1) consists of two separate networks, a terrestrial and a space network. The two networks are interconnected via several unique asynchronous transfer mode (ATM) virtual circuits (VCs) passing through an Adtech SX/14 (Spirent Communications, Rockville, MD) channel simulator. The SX/14 allows the insertion of time delays and random bit errors into the network flow. The networks on each side of this channel consist of a Cisco 7100 router (Cisco Systems, Inc., San Jose, CA) connected to the SX/14 via ATM and a Cisco Catalyst 2900 Ethernet switch connected to the routers via fast Ethernet. The Catalyst switches serve as our local area networks (LANs) connecting to transfer originators, receivers, or analyzers for the tests.

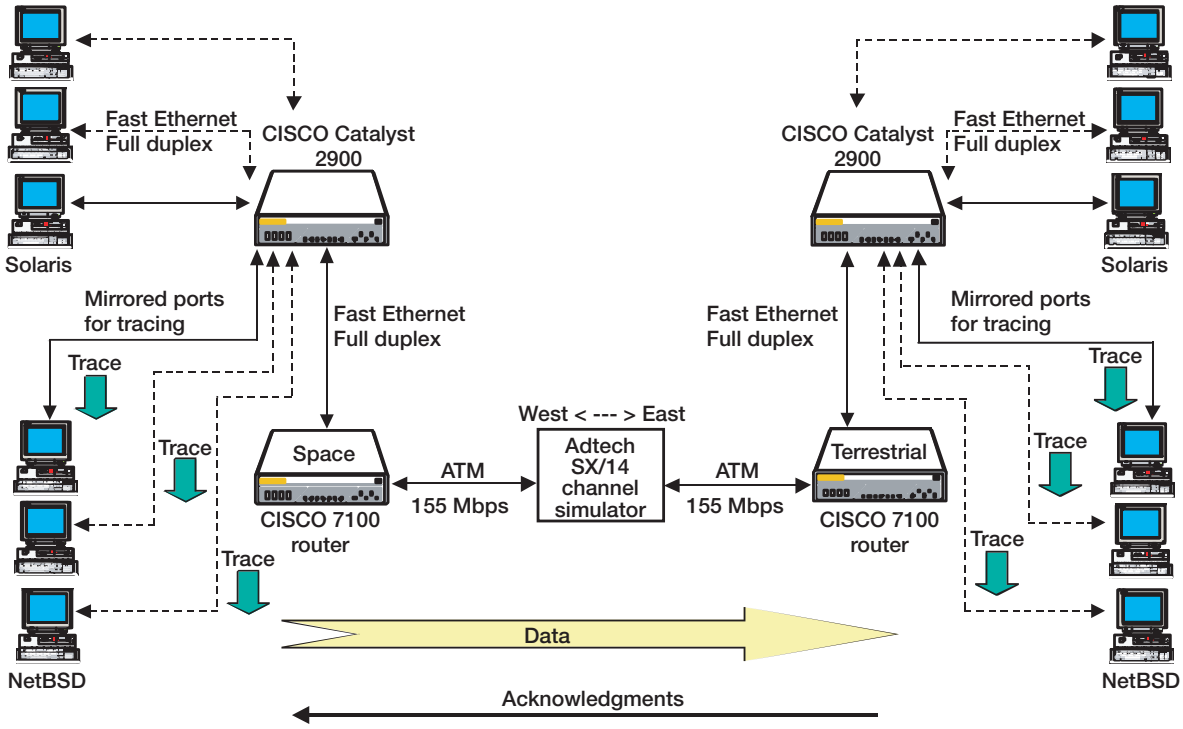


Figure 1.—Test bed facility. Dashed lines show additional machines used in multiple-flow tests.

Hosts on these switches are connected either to an active port to allow the system to participate in a traffic flow or to a mirrored port to allow the system to analyze the traffic to and from a specific host. In addition, hosts on each LAN are configured with two physical interfaces. The first interface is required for external access with the host on the test bed to avoid impeding traffic flows or analysis. The second interface on each host serves the tests and has multiple logical IP network interfaces with explicit route statements to force IP traffic to take a specific route through one of the aforementioned unique VCs interconnecting the test bed networks.

All analyses utilized well-known software programs such as “tcpdump,” “tcptrace,” and “xplot,” which were installed on the monitoring (tracing) machines running the NetBSD operating system as the main tools to capture, analyze, and observe the performance of our test protocols. The program tcpdump can be obtained from http://tcpdump.org, and tcptrace and xplot are both available at http://tcptrace.org. MITRE also modified the tcpdump and tcptrace, which we called “scps_tcpdump” and “scps_tcptrace,” that served as the tools for monitoring the SCPS–TP tests. It should be noted that a modified version of libpcap was needed from MITRE to compile with scps_tcpdump in order to use some options of tcpdump and debugging in SCPS.

Single-Flow Configurations

The single-flow tests provided a baseline comparison of congestion-based TCP versus SCPS–TP performance and an evaluation of rate-based protocols devoid of congestion-control mechanisms. During single-flow operation, two Sun machines (one on each network) running the Solaris 7 operating system were configured in full duplex mode, acting as either the receiver or the originator of a transfer session.

Transfers were between the terrestrial-single-rate and space-single-rate logical networks, forcing the single traffic flow to traverse our space channel simulator through a 100 Mbps fast Ethernet interface and a 155 Mbps ATM VC. To monitor and capture the network data, a PC running NetBSD 1.5 was connected to a mirrored port for each Sun system.

Software for the single-flow tests consisted of the following: the TCP with the Solaris 7 kernel that was used for the TCP–SACK test. The SCPS reference implementation (SCPS–RI versions 1.1.51, 1.1.62, and 1.1.66), provided by the MITRE Corporation, was used for the SCPS–TP tests. MDP testing utilized a Glenn-specific compilation of the MDP 1.9a4 source code. All detailed settings of TCP, SCPS–TP, MDP, and MFTP parameters for each test will be given in the **Testing Results** section and appendixes B through I. For baseline TCP testing, the popular "Test TCP Program" (ttcp), which was originally developed by the U.S. Army Ballistics Research Lab (BRL), was used to determine TCP performance. A version of ttcp modified by MITRE, called "scps_ttcp," was used as the benchmarking tool for the SCPS–TP performance. The scps_ttcp source is included in the SCPS–RI software package.

Single-flow testing procedures required a minimum of ten to thirty 1024-byte packet transfers for each series of tests in order to determine average protocol measurements and their deviations. A series consisted of manipulating BER (possible: 0, 10^{-8}, 10^{-7}, 10^{-6}, 10^{-5}, and 10^{-4}), round-trip transmission delay (possible: 0, 10, 250, and 500 ms), and file sizes (possible: 100 KB, 1 MB, 10 MB, 100 MB), creating up to 96 different series of tests to conduct. For this reason, scripts were created to automate the following tasks:

(1) Remotely set the BER and delay on the SX/14 channel simulator for each series of tests.
(2) Login to data collection systems via the secure shell protocol (ssh), initiating the appropriate data collection process.
(3) Login to the sender and receiver of a protocol transfer, initiating the file transfer.
(4) Monitor for completion of the transfer, stop collection processes, and save the data collected.
(5) Repeat steps (2) to (4) multiple times for each series of tests.

Additional series consisted of the many SCPS options exercised such as Van Jacobson congestion, Vegas congestion, Vegas corruption, pure rate with delayed ACKs, and pure rate with acknowledge every other packet for a total of up to 480 series of SCPS-related tests. Each series of tests consisted of 10 to 30 file transfers. The sheer number of tests to be performed for all evaluated protocols ran into over 4000 wall-clock hours of test bed utilization.

Because Glenn personnel were not working with thoroughly established and tested protocols, the results could not be taken at face value. Thus, a preliminary analysis of the collected data for each series of tests was necessary to ensure that the protocols were performing within the realm of expected behavior.

Multiple-Flow Configurations

The network architecture and procedures for multiple-flow testing are similar to that of the single-flow testing, with the exception that there are now three sender and three receiver machines on each side of our aforementioned terrestrial-space Internet. Each of these machines either initiates or receives a transfer session between the terrestrial-multiple-rate and space-multiple-rate logical networks, forcing all traffic to traverse our space channel simulator through an ATM VC, which is rate limited to a 15-Mbps bandwidth. This lower rate VC was used to ensure that congestion would occur. In addition to the active participant changes, three monitor machines were also required on each of our networks, because a machine using a mirrored port can only monitor traffic to and from one host in a switched environment. The dashed lines in figure 1 connect the additional machines needed for the multiple-flow tests.

In addition to the software used in previous testing for the TCP–SACK tests, the TCP included with the Solaris 7 kernel was used in the first two pairs of sender/receiver machines. Due to administrative constraints, the TCP included in the Solaris 8 kernel was used in the third pair of machines. SCPS–RI version 1.1.62 was used in the SCPS–VJ and SCPS-Vegas congestion multiple-flow tests.

As in single-flow testing, file transfers and data collection were performed with the average throughput and standard deviation calculated from the test results. Since there are three pairs of sending and receiving machines, there are six possible combinations of sending orders among the three senders. These combinations were picked randomly using the output of the Perl random number function. In addition, by using the same random function, each sender was also randomly started from 1 to 8 s apart from each other.

More information on each component of the single-flow and multiple-flow test beds is given in appendix J.

Defined Protocol Options

Single flow.—The following is a summary of the possible variables and options used in the TCP, SCPS–TP, MDP, and MFTP single-flow tests:

(1) File sizes: 100 KB, 1 MB, 10 MB, 100 MB
(2) Packet size: 1024 bytes (a 1472-byte packet size was used in the MFTP tests)
(3) Delays: 0, 10, 250, 500 ms
(4) BERs: 0 (baseline), 10^{-8}, 10^{-7}, 10^{-6}, 10^{-5}, 10^{-4}

TCP: The option chosen for the TCP single-flow tests was use of SACKs.
SCPS–TP: The following options were chosen for the SCPS single-flow tests:

(1) Van Jacobson congestion control with acknowledge every other packet (SCPS–VJ)
(2) Pure rate control with acknowledge every other packet (SCPS-pure rate, option F2)
(3) Pure rate control with strictly delayed ACKs (SCPS-pure rate, option F0)
(4) Vegas congestion control with acknowledge every other packet, assume congestion (SCPS-Vegas congestion)
(5) Vegas corruption control with acknowledge every other packet, assume corruption (SCPS-Vegas corruption)

MDP: The following options were chosen for the MDP multicast tests:

(1) Parity: no parity or forward error correction (server)
(2) File transfer: initial transfer of file (and repairs) only (server)
(3) Rate: set to 40 Mbps (server)
(4) Postprocessing of data: none (client)
(5) Archiving of data: client

MFTP: The following options were chosen for the MFTP multicast tests:

(1) Maximum datagram unit: set to default of 1472 (server)
(2) Network environment: bind test hosts to a unicast interface (client and server)
(3) File transfer: initial transfer of file (and repairs) only (server)
(4) Transfer timeout: set to 100 times the transfers initial pass

Multiple flow.—The following is a summary of the possible variables and options used in the TCP and SCPS–TP multiple-flow tests:

(1) File size: fixed at 50 MB
(2) Packet size: 1024 bytes
(3) BER: limited to 0, 10^{-7}, and 10^{-5}
(4) Round trip delay: fixed at 500 ms
(5) Intervals: random between start of competing flows

TCP: The option chosen for these tests was use of SACKs.
SCPS–TP: The options that were chosen for the SCPS multiple-flow tests are (1) Van Jacobson congestion control with acknowledge every other packet (SCPS–VJ) and (2) Vegas congestion control with acknowledge every other packet, assume congestion (SCPS-Vegas congestion).

Theoretical Throughput

In order to gain a feel for the overall performance of the various protocols, we calculated some theoretical bounds for TCP and pure rate-based protocols. In general, these calculations are for very large transfers that have reached a steady-state condition. These theoretical bounds are not indicative of small file transfers or small transactions that utilize a few packets such as command and control transactions. Appendixes K and L list the theoretical throughput for both congestion-based and rate-based protocols, respectively.

Congestion-Based Protocols

The maximum theoretical throughput for congestion-control-based protocols (TCP–SACK, SCPS–VJ) running over an errored link is calculated using equation (1) from Mathis (ref. 16):

$$\text{Bandwidth} = 0.93 \times \frac{\text{MSS}}{\text{RTT} \times \sqrt{p}} \tag{1}$$

where MSS is maximum segment size and p is packet error rate. Note, this equation assumes that the system has reached steady state (i.e. extremely large file transfers, not command and control transactions). In the present experiments, the user data packet size was set at 1024 bytes,[2] which was then used as MSS in equation (1).

For an error-free environment, the maximum throughput is equal to the receiver window size divided by the RTT, as given in equation (2):

$$\text{Maximum throughput} = \frac{\text{Window size}}{\text{RTT}} \tag{2}$$

Note that this equation assumes sufficiently large transfers such that the time the transfer spends in *slow start* is insignificant.

[2]The 1024 was determined to be close enough to model Ethernet packets and is the default used in Berkeley's Network Simulator (NS).

To count the effect of overhead on the throughput, it is assumed that the overhead is 58 bytes (20 bytes of TCP header, 20 bytes of IP header, and 18 bytes of Ethernet header). Thus, the maximum throughput is decreased by a factor of 1024/(1024+58) or 5 percent.

Rate-Based Protocols

As a first-order approximation for a rate-based protocol, the total transfer time of a file will equal the time needed to transmit the original packets of that file plus the time required to resend the dropped packets (if there are any dropped packets) and one round-trip time that is used for the three-way handshake at the beginning of a connection. The maximum throughput can be calculated by dividing the file size by the total transfer time of that file. It is assumed that every dropped packet can be fixed in the first retransmission. To include the effect of the overhead on the throughput, the maximum throughput is decreased by 1024/(1024+58) as used in the throughput calculation for congestion-based protocols. The following is the formula that was used to compute the throughput of pure rate-control protocol for error links:

$$\text{Throughput} = \frac{1024}{(1024+58)} \times \frac{\text{File size} \times 8}{\left[\left(\frac{\text{File size} \times 8 \times p}{R}\right) + \left(\frac{\text{File size} \times 8}{R}\right) + \text{RTT}\right]} \tag{3}$$

where throughput is in megabits per second, file size is in megabytes, p is packet error rate, R is user-specified rate or line rate in megabits per second, and the RTT is in seconds.

Figure 2 shows the theoretical throughputs for pure rate-based and TCP protocols. In this figure, data are acquired with the file size held constant at 100 MB while the delay (RTT) is varied from 10 to 500 ms. For the theoretical calculations, a transmission rate of 100 Mbps was used as that was the available bandwidth of the link. In addition, a packet size of 1024 bytes was used. Thus, figure 2 illustrates the theoretical upper bound for a rate-based protocol.

Figure 2 also shows the theoretical throughput of TCP for three delays. Notice that current deployments of TCP—in particular Reno and SACK—are dramatically affected by errors on the link. In addition, performance drops off rapidly in high-delay environments. Also, notice that a rate-base protocol will far outperform current deployments of TCP in error-prone, high-delay environments.

System and Protocol Implementation Problems

This section contains a chronological summary of the various problems encountered and their associated resolutions.

NetBSD

Originally, the NetBSD machines were to be used as the receiver and sender systems, but obtaining a valid TCP baseline with these systems was not possible. Some out-of-order packets were observed being sent from the sender during delays of 10 ms. Packet retransmissions were observed when delays were 250 ms or higher, which led to very poor throughput performance in delayed environments of 250 to 500 ms.

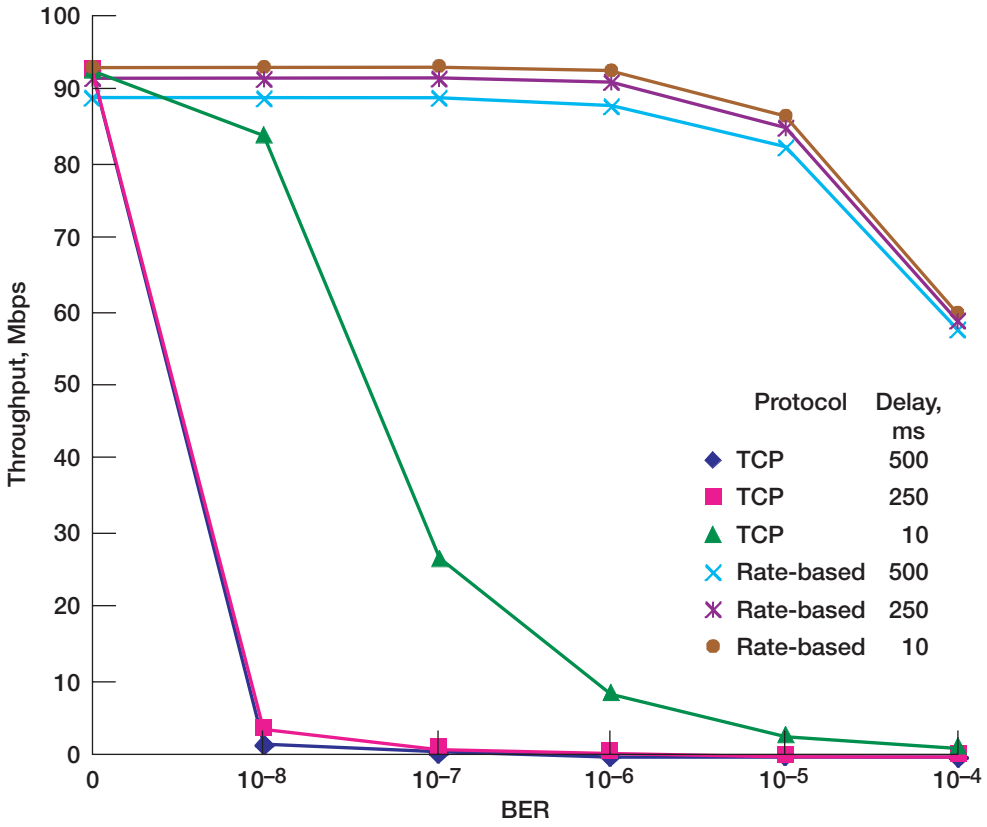


Figure 2.—Theoretical throughputs of TCP and rate-based protocols versus bit error rate (BER). Packet size, 1024 bytes; transmission rate, 100 Mbps. For rate-based protocols, file size was held constant at 100 MB.

In an attempt to improve performance of NetBSD TCP, parameters such as tcp_sendspace, tcp_recvspace, sb_max, and nmbcluster were increased. In addition, the network interface cards on the receiver and sender machines were changed from 100 to 10 Mbps. The throughput continued to be relatively low at high delays. Because these performance conflicts could not be resolved in a timely manner, two Sun Solaris machines were selected to be used as the sender and receiver systems since preliminary testing showed no such performance problems as compared with the NetBSD machines.

TCP and SCPS Tuning Problems

As indicated in the document "TCP Tuning Guide for Distributed Application on Wide Area Networks" (ref. 17), the optimal sender-receiver buffer size is calculated to be twice the bandwidth delay product (BDP). Therefore, initial tests were conducted in an effort to find the optimal receiver buffer size between one and two times the BDP.

Using the Sun machines, baseline tests were conducted for TCP–SACK, transferring 100-MB files in a zero BER environment with 250- and 500-ms RTT delays. During these tests, retransmitted packets occurred during the transfers when the receiver's buffer was set greater than the BDP and when the tests were executed back to back.

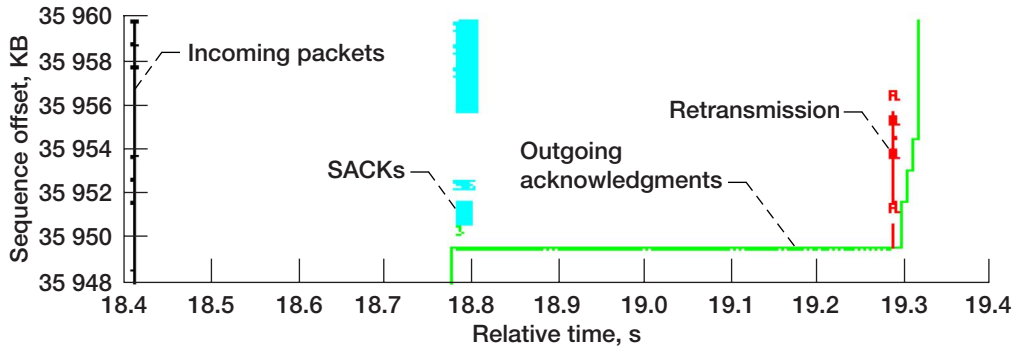


Figure 3.—Unnecessary selective acknowledgments (SACKs) that occured under TCP on receiving side.

In some cases, the retransmissions occurred in conjunction with the original segments missing in the receiver side tcpdump output, indicating that the packet was indeed lost. Most of the time, the receiver still generated the SACKs to trigger the retransmissions, even when the original packets appeared on the tracing machine interface of the receiving side (fig. 3). It is suspected that the receiving machines were unable to keep up with the high-speed transfers and that packets were dropped at the receiver interface or were processed incorrectly from the interface up to the transport layer.

While tuning the SCPS–TP at the line rate of 100 Mbps, unexpected retransmitted packets were also observed, which hindered the protocol abilities to operate at optimal levels. An exception to this observation was the SCPS–VJ protocol, which was able to support the 100-Mbps line rate speed without retransmissions.

Tuning Method

To obtain the best performance of TCP and SCPS–TP tests, 15 tests of 100-MB files were transmitted at the three delays. The values of the receiver buffer started at double the BDP, decreasing to about 90 percent of the BDP, which gave the highest average throughput and the lowest standard deviation.

As a result of the previously described tuning method, the following table gives values of the receiver buffer size used for the single-flow TCP and SCPS–TP tuning tests:

Delay, ms	Receiver buffer size, MB
500	6.25, 6.2, 6.1, 6.0, 5.9, 5.85, 5.8, 5.7
250	6.25, 3.125, 3.1, 3.0, 2.9, 2.85, 2.8, 2.7
10	0.250, 0.125, 0.12, 0.118, 0.116, 0.114

The optimal receiver buffer sizes and setting of other parameters for each individual testing set are provided later in the **Testing Results** section and in appendixes B through G.

SCPS-Vegas

While tuning for the SCPS-Vegas tests at a 500-ms delay, a 100-Mbps transmission rate, and a zero BER, it was occasionally observed that some transfers had retransmitted packets. This occurred when the receiver's buffers were set between 90 and 100 percent of the BDP. This phenomenon of retransmitted packets was observed until the SCPS–TP maximum rate option was lowered to 60 Mbps, which was then determined to offer the best performance for the SCPS-Vegas tests.

SCPS-Pure Rate

In the MITRE implementation of SCPS-pure rate, the sending rate in the testing depended on the set transmission rate and the receiver window. In the present tests, the sending rate did not respond correctly to the set transmission rate. In both SCPS-pure rate tests, the highest average throughput was always about 44 Mbps with the transmission rate set to 45 Mbps or greater. In addition, SCPS-pure rate did not have an optimum performance at a setting of 100 Mbps. Empirical results indicated that 80 Mbps was the maximum setting under which the MITRE implementation would operate properly. Therefore, a sending rate of 80 rather than 100 Mbps was used.

SCPS-Van Jacobson

While running the SCPS–VJ baseline tests, transfers using the Sun Solaris machines slowed down without taking any losses or having out-of-order packets. Figure 4 illustrates this throughput falloff in an error-free transfer (shown by a slight decrease in slope) 10 s into the connection. When using the NetBSD machines as the ttcp sender and receiver, a throughput of around 70 Mbps could be achieved while transferring 100-MB files at a 10-ms delay over an error-free link. We concluded that the problem, which may be related to the behavior of the Solaris operating system as it pertains to the SCPS protocol running outside the kernel, did not warrant using other systems for testing, though we acknowledge that further investigation of this phenomenon is indicated.

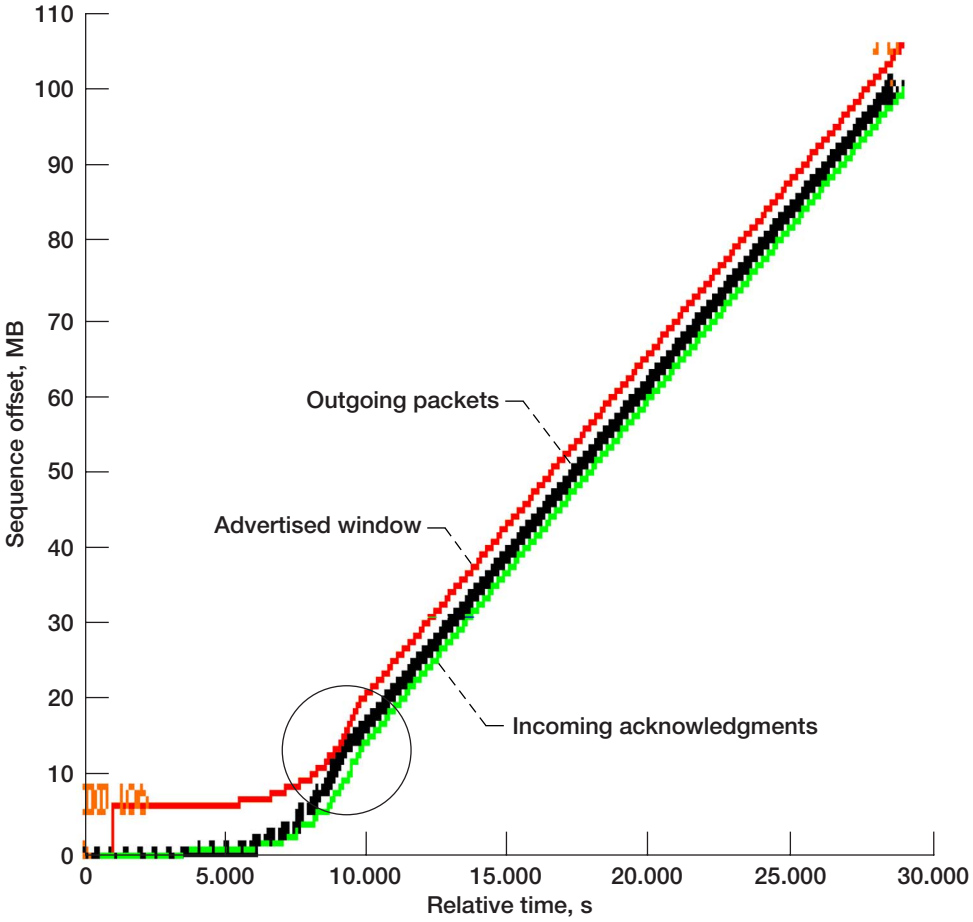


Figure 4.—Questionable throughput reduction in SCPS-Van Jacobson baseline tests on sending side. Circle highlights decrease in slope identifying reduced throughput in error-free transfer.

Multiple-Flow Testing

Initially, attempts were made to use a single pair of Sun Solaris sender-receiver machines for multiple-flow testing. In this method, several scps_ttcp processes were started between the sender and receiver, with each process using the same physical interface but directed to a different port on the receiving side by use of a dynamic port option, -p (scps_ttcp -p).

Unlike regular TCP using the ttcp program, the unmodified SCPS–RI code would always send a packet out of port 5001 from the sender. The regular TCP ttcp program would send the packet out of unique high port numbers with each invocation of the ttcp code. As a result, three copies of the scps_ttcp code were compiled, and each code had a different port number hard-coded into the compilation.

Unfortunately, further multiple-flow testing with these hard-coded port changes showed that load sharing of the three scps_ttcp receive and send processes on the one pair of sender and receiver machines took precedence, thus giving undesirable results for this emulation. Regardless of whether one or two of the three flows were still transmitting while the previous flow had completed, all three transmissions resulted in the same average throughput, which equated to one-third of the available bandwidth. This occurred even if the first transmission had ended a time before the second and third, or if the third was still transmitting while the first and second flows had completed. With fewer packets on the link to cause congestion, it was expected that the remaining executing flows would transmit more packets, but this was not the case on this single pair of machines. The eventual multiple-flow testing that was used for the final test results was then performed on three pairs of separate Sun sender-receiver machines, without the hard-coded port changes in the SCPS–RI code.

MDP

Initially setting the transmission rate above 4 Mbps caused the MDP receiver to crash when file sizes of 10 MB or greater were being sent. As a result, the MDP source code was obtained and recompiled with a MAXIMUM_RX_BUFFER set to 10 MB. After recompiling the MDP source, the MDP sender was able to send at rates up to 100 Mbps without crashing the receiver; however, the throughput performance at line rate was well below theoretical expectations.

Efforts to increase MDP performance included eliminating disk I/O by using a memory file system. When performance did not increase measurably, tcpdump files were examined to determine if the sender was sending at the user-specified rate. Further examination of the problem revealed that the receiver's resources were being overrun by the sender. Multiple-rate tests were conducted, which revealed that the maximum performance attained by MDP on the test systems was from a user-specified rate of 40 Mbps. Figure 5 illustrates the performance of the MDP throughput with rates from 20 to 80 Mbps.

For 100-MB transfers, MDP would often hang and would not complete the transfer. For these large files, MDP appeared to have its packet sequence numbers wrap around. We hypothesize that this may be the cause of the problem. This phenomenon requires further investigation.

Testing Philosophy

The goal of the testing was to evaluate a variety of transport protocols in a space-based environment (long-delay, error-prone links). However, for completeness, tests were conducted using RTT delays of 10 to 500 ms over the full range of BERs (0 to 10^{-4}). Note that a 10-ms delay is not indicative of a space-based environment but is indicative of a LAN. Thus, having a real-world situation where there are BERs of 10^{-7} or higher with 10-ms delays is unlikely unless one is considering a wireless LAN.

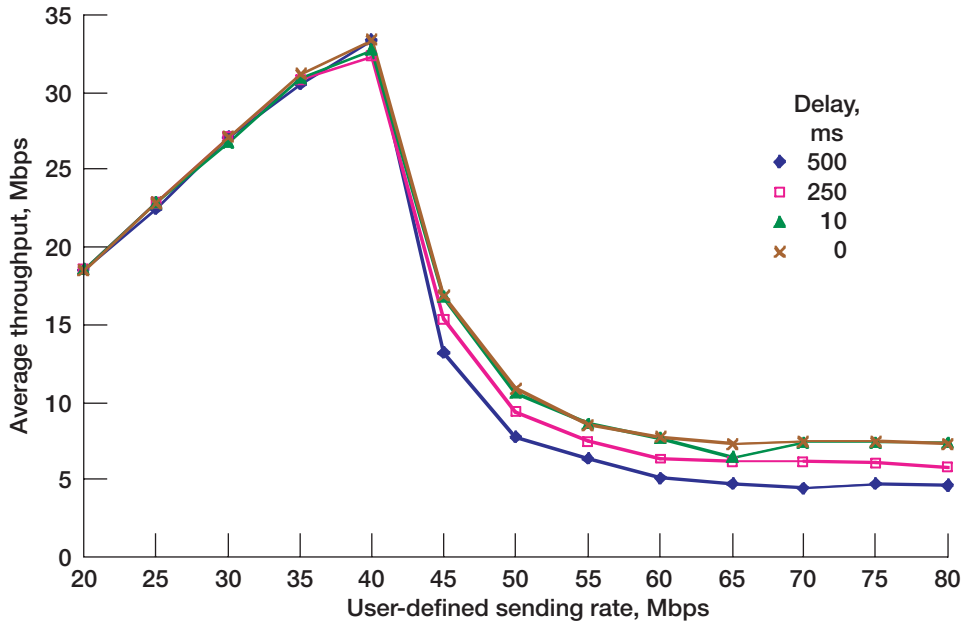


Figure 5.—Performance of Multicast Dissemination Protocol (MDP) versus rate. Bit error rate, 0; file size, 10 MB.

Although initial testing of the TCP and SCPS protocols under NetBSD yielded better performance for some configurations (see the section **Recommendations for Further Testing**), testing under Solaris produced results that more closely matched the theoretical characteristics of the test protocols. For these reasons, Solaris was selected as the test platform for all protocols.

During the initial investigations and baseline tests, it was noted that neither TCP nor SCPS was performing with a high degree of consistency when compared with theory or protocol specifications. Results obtained from testing ranged from optimal (theoretical) to suboptimal. It was concluded that data yielded from all testing should be deemed valid and should not be discarded. Instead, inconsistencies were thoroughly examined to determine if the suboptimal results were due to the protocol implementation, test strategy, or system (e.g., hardware and operating system) implementation. Suboptimal results that were due to testing problems or known protocol bugs were reconducted with the issue(s) addressed. In most cases, the most logical explanation for inconsistent results pointed to system implementation, which falls outside the scope of this group. As a result of having to deal with inconsistencies that were beyond control, the following testing philosophy evolved:

(1) Optimize and develop the baseline for the protocol on an error-free link for each BDP.
(2) Perform a minimum of 20 to 30 individual transfers for each series.
(3) Record all measurements (not just optimal runs).
(4) Capture and save all SYN and FIN packet traces and some dump files for each test series.
(5) Examine results as an entire series with inconsistencies due to controllable circumstances resulting in the reconducting of the entire series.

Testing Results

For each series of tests, 10 to 30 transfers were conducted to obtain the average transfer time and standard deviation. Time constraints and series complexity determined the number of tests for each series. Adhering to the above testing philosophy, average results with a large standard deviation were examined and reconducted if necessary. Transfer time and, subsequently, throughput results of each individual test were established dependant upon the protocol. In the case of TCP and SCPS, these statistics were determined from the tcpdump timestamp difference between the initial SYN and the final FIN captured on the sender side. Because MDP and MFTP do not use TCP three-way handshake and teardown, the transfer start was determined to be the time between the tcpdump timestamp of the first full length data packet seen from the sender and the timestamp of the first empty NACK seen from the receiver.

The results obtained from testing can be visualized in a variety of ways. Graphs depicting the overall performance of each protocol in the form of average throughput within controlled BER conditions are being provided throughout the rest of this section. Where applicable, transfers of different delays and file sizes are plotted to illustrate observations made, or conclusions drawn, as a result of testing. To make further comparisons, raw data from all tests conducted are available in appendixes B through I.

Single-Flow Congestion-Based Protocols

Configurations.—The following lists the configurations used for the single-flow tests of congestion-based protocols (TCP–SACK, SCPS–VJ, and SCPS-Vegas):

TCP–SACK: For the TCP–SACK test, the default values set on the Sun Solaris 7 kernel were used for most of the TCP/IP parameters, except for the following:

tcp_sack_ permitted: 2
tcp_wscale_always: 1
tcp_max_buf: 16777216
tcp_cwnd_max: 16777216
tcp_tstamp_if_wscale: 1
tcp_tstamp_always: 1

The definitions of these Sun TCP parameters are found at http://docs.sun.com/.

SCPS with Van Jacobson congestion control and acknowledge every other packet (SCPS–VJ): For the SCPS–VJ test, the default Delayed ACK timer was changed from 200 to 50 ms to be consistent with the TCP tests. The other SCPS–TP parameters were left unchanged. The SCPS rate option ("-R" option) was tried at both 80 and 100 Mbps. Since average throughput appeared a little higher at 100 Mbps, the SCPS rate of 100 Mbps was chosen for this testing set.

For both the TCP–SACK and SCPS–VJ, transfers using 100-MB files at BER 10^{-5} and below were not conducted. In addition, transfers using the other three file sizes (10 MB, 1 MB, 100 KB) were not conducted at a BER of 10^{-4}. It was determined that the excessive amount of time needed to conduct these transfers at the higher BERs would produce throughput results too uninteresting to warrant the time expended. Our preliminary results indicated that these transfers had a throughput of less than 0.5 percent of the baseline rate (e.g., less than 0.5 Mbps for 100 MB under both TCP and SCPS–VJ at a delay of 500 ms).

Appendixes B and C show the sample command, the optimum receiver buffer size values, the average throughput, and the standard deviation of the TCP–SACK and SCPS–VJ tests, respectively.

SCPS-Vegas congestion and SCPS-Vegas corruption with acknowledge every other packet: The only difference between SCPS-Vegas congestion and SCPS-Vegas corruption is that SCPS-Vegas congestion reduced *cwnd* by half in response to a dropped packet, whereas the *cwnd* in SCPS-Vegas corruption remained unchanged.

SCPS–RI version 1.1.62 was used for the SCPS-Vegas congestion tests, whereas version 1.1.66 was used for the SCPS-Vegas corruption tests. Version 1.1.66 is similar to version 1.1.62; however, version 1.1.66 had an additional switch option for the Vegas methods allowing either the original or modified *slow-start* mechanism as described in Vegas under SCPS from the **Protocol Overview** section.

As a result of a time constraint of the testing, only the Vegas tests of 10- and 100-MB files at BERs from 0 up to 10^{-5} were performed. The optimum rate settings for both Vegas tests were 60 Mbps at 500-ms delays and 80 Mbps at 250- and 10-ms delays.

Appendixes D and E show the sample command, the optimum receiving buffer size, average throughput, and standard deviation of the Vegas congestion and Vegas corruption tests, respectively.

Comparisons of SCPS-Vegas congestion, SCPS-Vegas corruption, TCP–SACK, and SCPS–VJ protocols.—Figure 6 shows the average throughput versus BER for 10- and 100-MB file sizes over a link with a 500-ms delay. Figure 7 shows the average throughput versus BER when transmitting a 10-MB file over various delays. In general, the overall characteristics of the three congestion-based protocols were similar. Both TCP–SACK and SCPS–VJ use the Van Jacobson congestion-control algorithm; thus, the overall characteristics of the SCPS–VJ are similar to those of TCP–SACK. SCPS-Vegas congestion performs slightly better then TCP–SACK. Notice that, in all cases, larger files have a better throughput at zero BER than do smaller files. This is because the slow-start algorithm has less an effect as files get larger. Also, notice that smaller files have better throughput at higher BERs than do larger files. For TCP–SACK and SCPS–VJ, the better throughput is due to the additive increase, multiplicative decrease congestion-control algorithms.

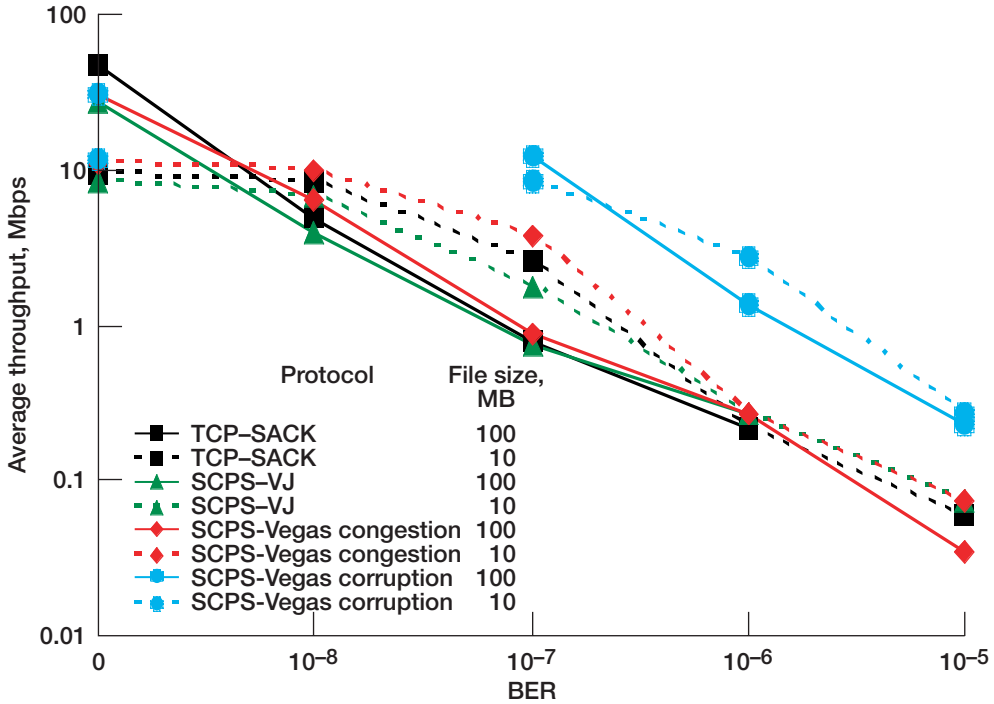


Figure 6.—Performance of congestion-friendly protocols versus bit error rate (BER) for 10- and 100-MB file sizes. Delay, 500 ms.

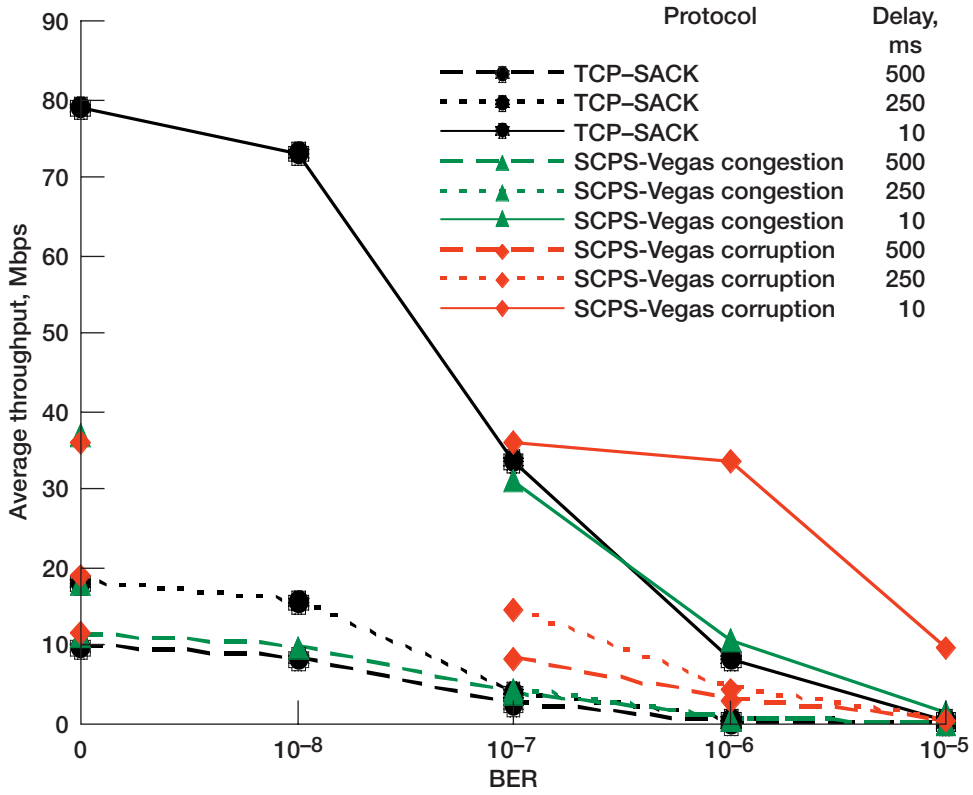


Figure 7.—Performance of congestion-friendly protocols versus bit error rate (BER) for 10-MB file size.

Since the *cwnd* is cut in half each time an error occurs in the SCPS-Vegas congestion transfers, their average throughput declined faster when errors were inserted in the network than that of SCPS-Vegas corruption transfers. Furthermore, SCPS-Vegas corruption performed better than all congestion-based protocols at higher BER, but still did not perform as well as a pure rate-based protocol as congestion control was still being performed.

Single-Flow Rate-Based Protocols

Since there is no congestion-control mechanism involved in these tests, pure rate-control performance tests can only be conducted on a network devoid of competing flows. For this reason, tests were conducted as a single flow, one transfer at a time. Testing in a multiple-flow environment would be inappropriate.

Configurations.—The following lists the configurations used for the single-flow tests of rate-based protocols (SCPS-pure rate, MDP, and MFTP):

SCPS-pure rate with acknowledge every other packet (SCPS-pure rate-F2) and SCPS-pure rate with strictly delayed ACKs (SCPS-pure rate-F0): For the SCPS-pure rate-F2 tests, SCPS–RI version 1.1.51 was used with the Delayed ACK timer changed to 50 ms. The optimum rate for this test set was 80 Mbps as determined through the previously described tuning tests.

However, while tuning for the SCPS-pure rate-F0 tests using the SCPS–RI 1.1.51 version, it was noticed that the ACKs were not being sent back every 200 ms as defined by the default Delayed ACK Timer. Instead, the ACKs were sometimes being sent much later than 200 ms. The shortest time for the ACKs returning back to the sender was 200 ms, and the longest time was 1 to 2 s. This reduced rate of returning of ACKs led to a receiver window-limiting problem because the receiver window was not updated fast enough. Therefore, a newer version of the SCPS–RI was used, version 1.1.62, which did

send the ACKs correctly per the defined Delayed ACK timer. As a result, the Delayed ACK timer for the SCPS-pure rate-F0 tests was left unchanged at the default of 200 ms using the new SCPS–RI 1.1.62. Like the SCPS-pure rate-F2 tests, 80 Mbps was also determined to be the optimum rate for the SCPS-pure rate-F0 test.

Appendixes F and G show the sample command, the optimum receiver buffer size values, average throughput, and standard deviation of the SCPS-pure rate-F2 and SCPS-pure rate-F0 tests, respectively.

MDP: Systems involved in the MDP were configured to use their unicast address at an optimal rate of 40 Mbps. The server was set to send data blocks without parity (forward error correction) and to retransmit its payload for an indefinite number of passes (server resends repair data until an empty NACK is received). The receiver was configured to archive the data and to refrain from any postprocessing (the default was to open the data in a Web browser) as this was the only way to get the MDP receiver to log a file transfer completion.

MFTP: Systems involved in the MFTP transfers were configured at the application maximum rate of 50 Mbps. In addition, a unicast address was specified for both systems. The MFTP server was set to transmit the file only once but to transmit repair data in answer to client NACKS either for up to one hundred times the initial transfer time or until the client signals completion with an empty NACK.

Comparisons of SCPS-pure rate-F2, SCPS-pure rate-F0, MDP, and MFTP protocols.—Figure 8 shows the performance of the various rate-based protocols for a 500-ms RTT delay. None of the rate-based protocols tested meets the theoretical throughput when large files are transmitted. This result may be due to the capabilities of the machines in combination with implementation issues. Although the experimental throughput results of all rate-based protocols were lower than the theory, the 10-MB MFTP curves at the three delays matched closest the shape of the theoretical curves. Under all implemented BER and DELAY conditions, MDP performed well up to approximately 35 Mbps and slightly better than both of the SCPS–TP pure rate protocols (see appendix H). Figure 5 illustrates an MDP performance falloff possibly due to receiver-processing problems at higher transmission rate settings. In fact, for 100-MB files, MDP would not complete the transfer because the receiver would choke (possibly due to the packet number wraparound phenomenon previously discussed in the section **MDP** under **System and Protocol Implementation Problems**). The SCPS-pure rate protocol also fell off faster than predicted at

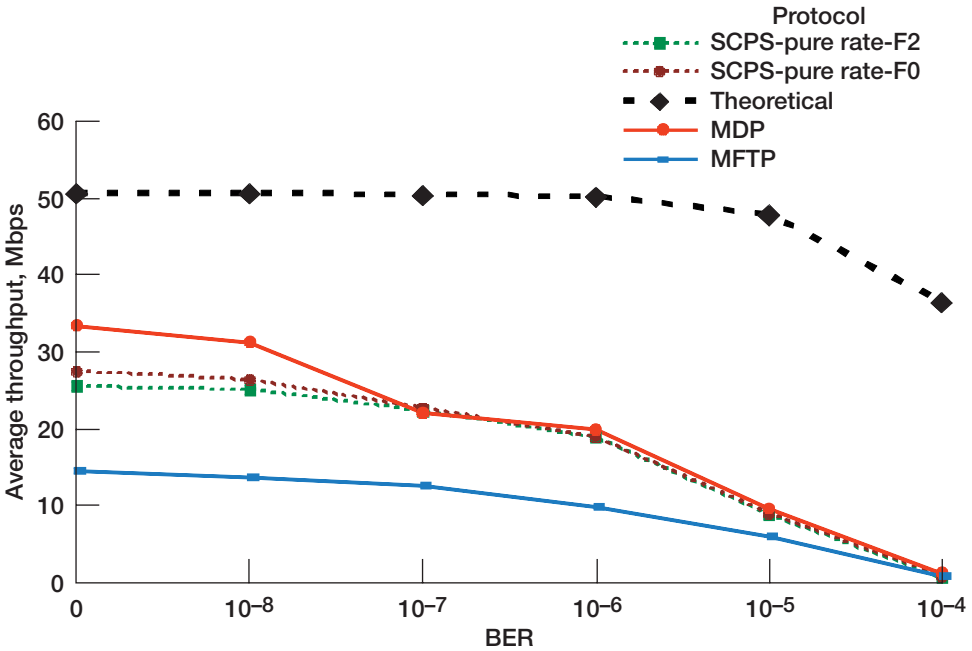


Figure 8.—Performance of rate-based protocols versus bit error rate (BER) for 10-MB file sizes. Delay, 500 ms.

the high BERs and did not perform anywhere near theoretical throughput, even at low BERs. As with MDP, this may have been caused by the receiver becoming overwhelmed and unable to keep pace with the sender—particularly at higher BER and high delay.

Note in figure 9 that the SCPS-pure rate-F2 and -F0 curves at a 10-ms delay closely follow the shape of the corresponding theoretical curves as compared with these curves at 500-ms delays. Thus, we are led to believe that the test machines may have a memory management problem. In addition, an in-kernel implementation of SCPS may improve performance (refs. 18 and 19).

Multiple-Flow Results

The multiple-flow tests were designed to exercise the congestion-control properties of the congestion-based protocols. In the single-flow tests, an individual flow could utilize the entire bandwidth. In the multiple-flow tests, three pairs of flows competed for the available bandwidth. For these tests, the bandwidth was set to 15 Mbps at the ATM router interfaces. Time constraints allowed only a limited subset of tests to be performed with BERs of 0, 10^{-7}, and 10^{-5}, an RTT delay of 500 ms, and a single file size of 50 MB. In addition, SCPS-Vegas corruption was not tested in this environment, as it would have been a misapplication of the protocol.

Similar to the single-flow tests, SCPS-Vegas congestion performed slightly better than TCP–SACK, which performed slightly better than SCPS–VJ at zero and moderate BERs, with an RTT delay of 500 ms (fig. 10). At a BER of 10^{-7} and 10^{-5}, the throughput of each pair in multiple-flow tests had almost the same performance as in the single-flow tests under these same error conditions. The reason was that the packet loss due to errors dictates the performance rather than actual congestion. Notice that the total average throughput can exceed the network capacity because the random offset start times for the three flows (where the flows are all started at random intervals) cause individual flows to transfer and complete during different usages of the available bandwidth. The total average throughput of any one set of tests can exceed 15 Mbps, particularly if the first and last transfers do not overlap by much.

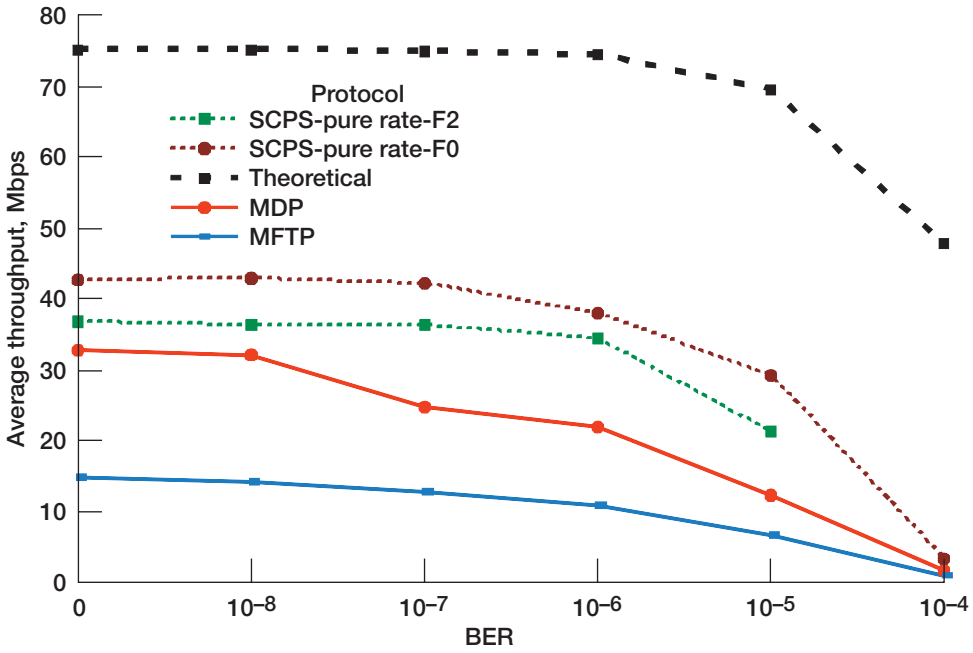


Figure 9.—Performance of rate-based protocols versus bit error rate (BER) for 10-MB file size. Delay, 10 ms.

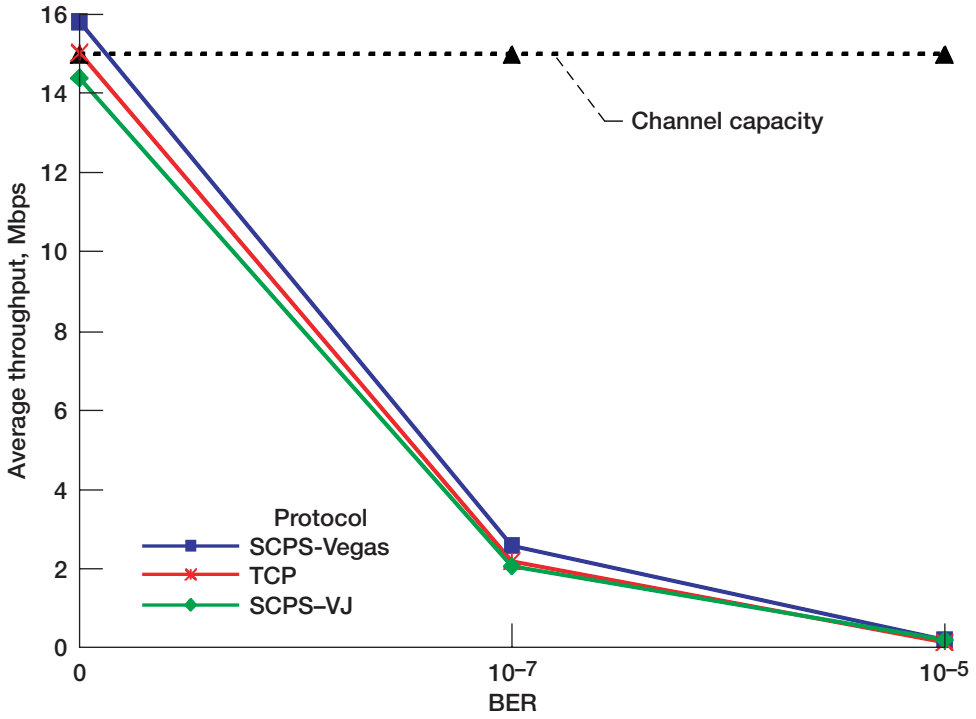


Figure 10.—Performance of multiple-flow protocols versus bit error rate (BER). Delay, 500 ms.

Figure 11 shows the throughput of each TCP–SACK flow in individual runs for all 30 tests. Tests reflect each sender and receiver pair of machines at zero BER. Similar results were achieved for the SCPS–VJ and SCPS-Vegas congestion (see appendix M for all three graphs). A detailed analysis of the individual data runs shows that no flow had a throughput below 3 Mbps in all 30 test runs of SCPS-Vegas congestion. However, in the TCP–SACK and SCPS–VJ tests, 20 out of the 30 test runs have a throughput below 3 Mbps. Although the TCP–SACK and SCPS–VJ tests have throughput rates of 2.3 through 7.9 Mbps and 1.9 through 7.5 Mbps, respectively, the minimum and maximum throughputs of the flows in the SCPS-Vegas congestion tests are 3.2 and 9.5 Mbps.

Although not shown in the graphics, it is important to note that at a BER of zero, the percentage of retransmitted packets in SCPS-Vegas corruption was almost none as compared with 0.3 and 0.16 percent in TCP–SACK and SCPS–VJ, respectively. The improvement occurs because under SCPS-Vegas congestion control, the *cwnd* is monitored between the *alpha* and *beta* thresholds.

Recommendations for Further Testing

During the preliminary testing, 100-MB files were transferred at 100 Mbps with a 10-ms delay using an error-free link between two NetBSD machines. The throughputs of these SCPS–VJ and TCP–SACK tests were around 70 and 79 Mbps, respectively. However, while running the same test for SCPS–VJ and SCPS-Vegas congestion using Solaris machines as sender and receiver, the maximum average throughput was about 38 Mbps, which was much lower than the 87 Mbps for TCP–SACK on Solaris. Furthermore, with the Solaris sender and receiver under the 100-Mbps link capacity, none of the SCPS-pure rate transfers could achieve a throughput higher than 44 Mbps, despite the SCPS setting rate. This low throughput of the SCPS transfers may possibly be caused by the Solaris operating system, which may have some effect on processes running outside the kernel, such as SCPS–TP. This low throughput needs further investigation.

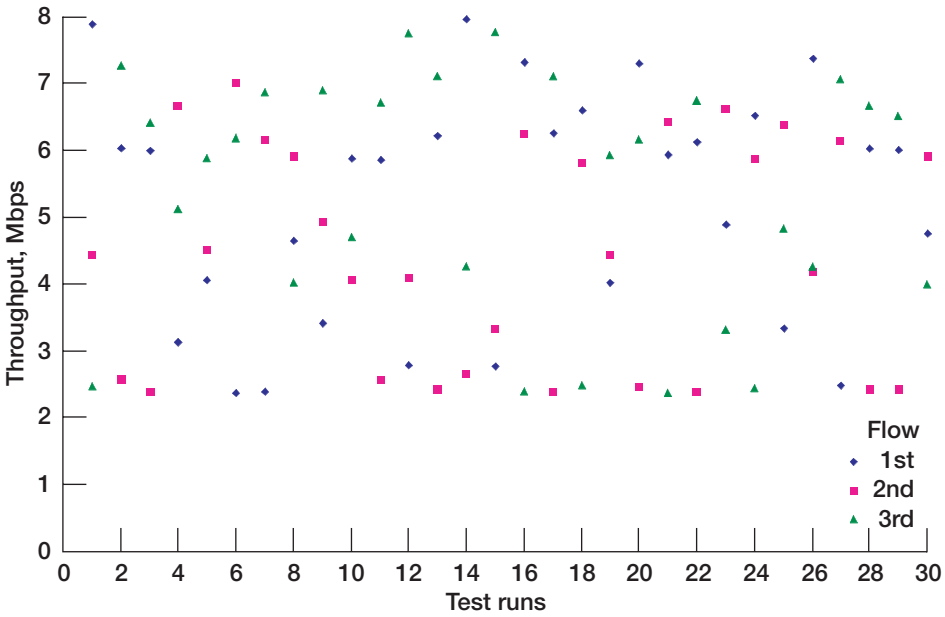


Figure 11.—Performance of TCP–SACK for multiple-flow transfers.

Also, with the zero-BER and 500-ms delay link, the throughput of a 100-MB file transfer under SCPS-pure rate-F2 using a NetBSD sender and a Solaris receiver does not respond correctly to the SCPS setting rate when the rate is set lower than 40 Mbps. The throughput was 14 Mbps when the SCPS rate option was set at 30 Mbps, even though there were no dropped packets in the transfer. On the other hand, the same transfer between two Solaris machines gave a throughput of 27 Mbps (with the SCPS rate set to 30 Mbps). The reason behind this difference needs further investigation.

All rate-based protocols need further testing on faster machines and with different operating systems to determine whether those changes will improve performance or if these protocols need further development. For the commercial rate-based protocols tested, the poor performance may be due to the algorithms and coding being optimized for multicast operation. Therefore, an investigation of SCPS performance with different operating systems using faster machines is suggested. Also, using an in-kernel SCPS–TP implementation may lead to better performance of SCPS in an error-free environment.

According to reference 20, the SCPS-pure rate protocols utilize the TCP header and advertise themselves as TCP via the protocol number field in the IP header. However, SCPS-pure rate is not performing congestion control. The authors suggest that, for such operation, SCPS–TP should advertise a different protocol number to ease quality-of-service provisioning. Failure to do so may result in SCPS-pure rate flows dominating shared links or being identified as rogue sources.

SCPS-Vegas should be further investigated for operation in mobile environments and for performance on intermittent links as the Vegas algorithm has some known problems (refs. 20 and 21).

Vegas uses estimates of the propagation delay and base RTT to adjust its window size. Thus, it is very important for a TCP-Vegas connection to have accurate estimates of these quantities. Rerouting a path may change the propagation delay of the connection, which could result in a substantial decrease in throughput.

Each TCP-Vegas connection attempts to keep a few packets in the network. When the estimation of the propagation delay is off, the connections could inadvertently keep many more packets in the network, causing a persistent congestion.

Flows using the Vegas congestion-avoidance algorithm intentionally lower their transmission rate under heavy congestion, assuming that they are the sole source of congestion and can fix the problem. Thus, in head-to-head transfers, TCP-Reno steals bandwidth from Vegas, which is one possible reason why Vegas has not seen wide deployment in the Internet.

Conclusions

There have been numerous discussions relative to the role of the Space Communications Protocol Standards Transport Protocol (SCPS–TP) in the ever-evolving Transmission Control Protocol (TCP) family, particularly with respect to other known TCP features that accommodate high-bandwidth and long-delay networks (e.g., large windows and selective acknowledgments). To gain a better understanding of the performance of the SCPS–TP relative to other TCP variants and to determine the maturity of the various options for use on higher rate space links, the NASA Space Communications and Data Systems (SCDS) Office requested that Glenn Research Center perform a comprehensive set of tests. The goals were to validate the operation of the reference implementation of the SCPS–TP relative to its controlling Consultative Committee for Space Data Systems (CCSDS) specification and to perform a comprehensive comparison of SCPS–TP options relative to other TCP options.

We have studied the effect of delay and bit error rate (BER) on the performance of congestion-based and rate-based protocols in emulated space links under uncongested conditions and with limited congestion. The results correlate well with previous testing of TCP options, including the SCPS–TP: The single-flow and multiple-flow test results clearly illustrate that the TCP-Vegas enhancements to TCP (implemented by the SCPS–TP) provide performance improvements over a TCP–SACK (selective acknowledgment) implementation tested in high-bandwidth-delay-product environments. Since performance considerations are subjective, the operational value of these performance increases can best be assessed by the users of specific applications hosted within those environments.

Very small transactions such as command and control will probably see little difference in performance for any variant of TCP.

In extremely error-prone environments with high round trip time (RTT) latencies, use of a rate-based TCP variant such as that provided by SCPS–TP is advisable, assuming that the network implementation is properly engineered. Users are cautioned against using rate-based protocols on networks where resource allocation policies are based on contention.

The tested SCPS–TP implementation was the protocol reference implementation, which exists as a user application rather than as a more generalized protocol service provider. For some deployments, an in-kernel protocol service provider may prove more desirable than the deployment of application-level protocol service providers. The tradeoff is between the more efficient use of resources and possibly higher performance for in-kernel services versus the ease of maintainability and deployment of application-level implementations. Commercial in-kernel and application-level implementations of SCPS–TP exist but have not been tested as part of the present effort.

Even with equal performance, the SCPS–TP version of a rate-based protocol may be more desirable to implement than other rate-based protocols (such as the Multicast Dissemination Protocol (MDP)) since SCPS–TP may require only sending-side modification. This feature of the SCPS–TP eliminates many of the risks associated with requiring universal deployment of new software.

Existing TCP implementations (drawn from a variety of communities including the SCPS–TP) appear to satisfy all currently known space mission needs; however, the space mission community should maintain an awareness of current and future TCP research being performed by other communities.

Appendix A
Acronyms

ACK	acknowledgment
ATM	asynchronous transfer mode
BDP	bandwidth delay product
BER	bit error rate
BRL	Ballistics Research Lab
CCSDS	Consultative Committee for Space Data Systems
FP	file protocol
IETF	Internet Engineering Task Force
IP	Internet protocol
LAN	local area network
MBONE	multicast backbone
MDP	Multicast Dissemination Protocol
MFTP	multicast file transmission protocol
MSS	maximum segment size
NACK	negative acknowledgment
NP	network protocol
RI	reference implementation
RTO	retransmission timer timeout
RTT	round trip time
SACK	selective acknowledgment
SCDS	Space Communications and Data Systems
SCPS	Space Communications Protocol Standards
SNACK	selective negative acknowledgment
SP	security protocol
TCP	Transmission Control Protocol
TP	transport protocol
UDP	User Datagram Protocol
VC	virtual circuit
VJ	Van Jacobson

Appendix B
TCP–SACK Single-Flow Test Results

The following table presents the average throughput and standard deviation for TCP–SACK single-flow test runs:

File size	Average throughput, Mbps[a,b]				
	Bit error rate, BER				
	0	10^{-8}	10^{-7}	10^{-6}	10^{-5}
Delay, 500 ms[c] [Buffer size, 5.7 MB]					
100 MB	47.4834 (0.8395)	4.9065 (4.2973)	0.7771 (0.0709)	0.2164 (0.0378)	N/A
10 MB	9.5483 (0.3027)	8.3500 (2.2326)	2.6268 (2.6236)	.6826 (1.7320)	0.0584 (0.0120)
1 MB	1.3536 (0.0645)	1.3410 (0.1040)	1.0424 (0.3336)	.3825 (0.2942)	.0474 (0.0057)
100 KB	.2060 (0.0406)	.2043 (0.0144)	.1987 (0.0249)	.1636 (0.0362)	.0597 (0.0307)
Delay, 250 ms [Buffer size, 2.85 MB]					
100 MB	63.4859 (0.5848)	9.4997 (4.9435)	1.4394 (0.1115)	0.4263 (0.0391)	N/A
10 MB	18.1965 (0.5912)	15.6319 (4.1887)	3.9613 (4.4278)	.4411 (0.0504)	0.0850 (0.0040)
1 MB	2.5998 (0.1032)	2.5732 (0.1867)	2.2223 (0.4789)	.7674 (0.5642)	.0928 (0.0210)
100 KB	.3933 (0.0243)	.4070 (0.0314)	.4011 (0.3400)	.3331 (0.0870)	.1113 (0.0710)
Delay, 10 ms [Buffer size, 250 KB]					
100 MB	87.4791 (0.1149)	77.9525 (3.8425)	29.8685 (1.6645)	8.2267 (0.2115)	0.6163 (0.0724)
10 MB	78.9545 (0.6349)	73.1726 (9.3939)	33.4123 (11.0870)	8.2680 (0.9914)	.6127 (0.1113)
1 MB	37.3901 (4.2840)	36.9025 (4.3958)	34.8699 (8.4956)	12.3240 (8.8622)	.4825 (0.2870)
100 KB	6.6980 (0.2096)	6.7909 (0.3810)	6.7904 (0.2680)	5.3115 (2.2524)	.4434 (0.8506)

[a]N/A indicates the test was not run.

[b]Numbers in parentheses are standard deviations.

[c]Sample session: Receiver: ttcp -b 5700000 -r -s
Sender: ttcp -1 1024 -t destination < input file

Appendix C
SCPS-Van Jacobson Single-Flow Test Results

The following table presents the average throughput and standard deviation for the SCPS-Van Jacobson single-flow test runs:

File size	Average throughput, Mbps[a,b]				
	Bit error rate, BER				
	0	10^{-8}	10^{-7}	10^{-6}	10^{-5}
Delay, 500 ms[c] [Buffer size, 6 MB; rate, 100 Mbps]					
100 MB	27.4214 (0.1557)	4.0033 (2.5455)	0.7384 (0.0463)	0.2663 (0.0043)	N/A
10 MB	8.5507 (0.0551)	6.9034 (2.4210)	1.7278 (1.5361)	.2729 (0.0175)	0.0731 (0.0028)
1 MB	1.2540 (0.0450)	1.2407 (0.0816)	1.0615 (0.2778)	.3829 (0.2697)	.0748 (0.0075)
100 KB	.2265 (0.0003)	.2031 (0.0649)	.2031 (0.0649)	.1766 (0.0522)	.0803 (0.0324)
Delay, 250 ms [Buffer size, 2.85 MB; rate, 100 Mbps]					
100 MB	31.4218 (0.1209)	6.4877 (2.7860)	1.4718 (0.0953)	0.5220 (0.0097)	N/A
10 MB	15.2065 (0.0879)	13.5034 (2.0712)	2.4579 (2.3595)	.5256 (0.0331)	0.1452 (0.0048)
1 MB	2.4909 (0.0842)	2.3900 (0.3145)	1.9173 (0.6706)	.7711 (0.5051)	.1495 (0.0158)
100 KB	.4488 (0.0012)	.4397 (0.0499)	.4437 (0.0208)	.3712 (0.0912)	.1729 (0.0669)
Delay, 10 ms [Buffer size, 250 KB; rate, 100 Mbps]					
100 MB	37.9841 (0.3258)	35.9497 (0.7486)	25.0959 (1.0451)	9.1357 (0.2251)	1.5178 (0.0417)
10 MB	36.6824 (0.3074)	35.1153 (1.5163)	24.7398 (3.3316)	9.2693 (0.4964)	1.5056 (0.1923)
1 MB	27.6818 (0.2990)	27.2875 (1.3681)	23.4541 (5.2481)	9.2660 (0.8087)	1.6729 (0.7189)
100 KB	7.9576 (0.2007)	7.9651 (0.2441)	7.7838 (0.6102)	6.8672 (1.6445)	2.6383 (1.1827)

[a]N/A indicates the test was not run.

[b]Numbers in parentheses are standard deviations.

[c]Sample session: Receiver: scps_ttcp -b 6000000 -G 1 -F 2 -R 100000000 -f m -r -s
Sender: scps_ttcp -b 6000000 -1 1024 -G 1 -F 2 -R 100000000 -f m -t
destination < input file

Appendix D
SCPS-Vegas Congestion Single-Flow Test Results

The following table presents the average throughput and standard deviation for the SCPS-Vegas congestion single-flow test runs:

File size	Average throughput, Mbps[a,b]				
	Bit error rate, BER				
	0	10^{-8}	10^{-7}	10^{-6}	10^{-5}
Delay, 500 ms[c] [Buffer size, 5.65 MB; rate, 60 Mbps]					
100 MB	30.1087 (0.2878)	6.3457 (3.3689)	0.8765 (0.0592)	0.2643 (0.0074)	0.0341 (0.0030)
10 MB	10.9610 (0.1173)	9.9182 (1.9097)	3.8187 (3.4269)	.2740 (0.0208)	.0746 (0.0025)
1 MB	1.5742 (0.0005)	1.5651 (0.0348)	N/A	N/A	N/A
100 KB	.2248 (0.0124)	.2269 (0.0100)	N/A	N/A	N/A
Delay, 250 ms [Buffer size, 2.85 MB; rate, 80 Mbps]					
100 MB	33.6895 (0.2368)	N/A	1.7959 (0.1632)	0.5232 (0.1419)	N/A
10 MB	18.0510 (0.0525)	N/A	4.5988 (4.1705)	.5341 (0.4150)	0.1426 (0.0056)
1 MB	N/A	N/A	N/A	N/A	N/A
100 KB	N/A	N/A	N/A	N/A	N/A
Delay, 10 ms [Buffer size, 120 KB; rate, 80 Mbps]					
100 MB	38.4582 (0.5183)	N/A	31.1326 (1.1599)	10.5089 (0.2098)	1.2914 (0.0493)
10 MB	36.8993 (0.5215)	N/A	31.1639 (2.6806)	10.5712 (0.7652)	1.3162 (0.1795)
1 MB	N/A	N/A	N/A	N/A	N/A
100 KB	N/A	N/A	N/A	N/A	N/A

[a]N/A indicates the test was not run.

[b]Numbers in parentheses are standard deviations.

[c]Sample session: Receiver: scps_ttcp -b 5650000 -G 2 -F 2 -R 60000000 -f m -r -s
Sender: scps_ttcp -b 5650000 -l 1024 -G 2 -F 2 -R 60000000 -f m -t destination <input file

Appendix E
SCPS-Vegas Corruption Single-Flow Test Results

The following table presents the average throughput and standard deviation for the SCPS-Vegas corruption single-flow test runs:

File size	Average throughput, Mbps[a,b]				
	Bit error rate, BER				
	0	10^{-8}	10^{-7}	10^{-6}	10^{-5}
Delay, 500 ms[c] [Buffer size, 6 MB; rate, 60 Mbps]					
100 MB	30.1282 (0.2766)	N/A	12.0248 (4.9803)	1.3793 (0.1140)	0.2300 (0.0265)
10 MB	11.7970 (0.0122)	N/A	8.4308 (1.9971)	2.7188 (1.1281)	.2604 (0.0748)
1 MB	N/A	N/A	N/A	N/A	N/A
100 KB	N/A	N/A	N/A	N/A	N/A
Delay, 250 ms [Buffer size, 2.9 MB; rate, 80 Mbps]					
100 MB	33.1291 (0.3359)	N/A	21.8444 (6.5917)	2.1451 (0.1466)	N/A
10 MB	19.0337 (0.0590)	N/A	14.7844 (3.2956)	4.2148 (1.7377)	0.4829 (0.0304)
1 MB	N/A	N/A	N/A	N/A	N/A
100 KB	N/A	N/A	N/A	N/A	N/A
Delay, 10 ms [Buffer size, 125 KB; rate, 80 Mbps]					
100 MB	36.8928 (0.3441)	N/A	36.5510 (0.4042)	34.3871 (2.0728)	9.3872 (0.9783)
10 MB	36.1105 (0.2342)	N/A	35.7929 (0.2733)	33.6843 (0.5175)	9.7608 (1.0086)
1 MB	N/A	N/A	N/A	N/A	N/A
100 KB	N/A	N/A	N/A	N/A	N/A

[a]N/A indicates the test was not run.

[b]Numbers in parentheses are standard deviations.

[c]Sample session: Receiver: scps_ttcp -b 6000000 -G 2 -F 2 -R 60000000 -f m -r -s
Sender: scps_ttcp -b 6000000 -l 1024 -G 2 -F 2 -R 60000000 -f m -t
destination <input file

Appendix F
SCPS-Pure Rate With Acknowledge Every Other Packet Single-Flow Test Results

The following table presents the average throughput and standard deviation for the SCPS-pure rate with acknowledge every other packet single-flow test runs:

File size	Average throughput, Mbps[a]				
	Bit error rate, BER				
	0	10^{-8}	10^{-7}	10^{-6}	10^{-5}
Delay, 500 ms[b] [Buffer size, 5.9 MB; rate, 80 Mbps]					
100 MB	35.3181 (0.2454)	34.2821 (0.3535)	31.3510 (0.4056)	26.4681 (0.6770)	11.3075 (0.3628)
10 MB	25.4876 (0.1537)	24.9645 (0.7979)	22.2588 (1.0536)	18.7849 (0.1843)	8.8789 (1.0768)
1 MB	6.6126 (0.0092)	6.4950 (0.4571)	5.8022 (0.8835)	4.5472 (0.5447)	2.2120 (0.5564)
100 KB	.7782 (0.0009)	.7685 (0.0456)	.7447 (0.0871)	.6478 (0.1345)	.3976 (0.1483)
Delay, 250 ms [Buffer size, 2.9 MB; rate, 80 Mbps]					
100 MB	36.0155 (0.3341)	35.7043 (0.5609)	32.7728 (1.4272)	29.0794 (0.4985)	14.6034 (0.4478)
10 MB	30.1495 (0.2929)	29.5398 (0.8830)	26.9850 (1.0908)	23.8974 (1.2606)	11.1713 (2.6557)
1 MB	11.2557 (0.2180)	10.9805 (0.8008)	10.4473 (1.0045)	8.3564 (0.5084)	3.9626 (1.7046)
100 KB	1.5112 (0.0261)	1.4851 (0.1670)	1.5144 (0.0030)	1.2140 (0.2322)	.7618 (0.2644)
Delay, 10 ms [Buffer size, 125 KB; rate, 80 Mbps]					
100 MB	37.8917 (0.3564)	37.5283 (0.5107)	37.5071 (0.3737)	35.3899 (0.4316)	23.6980 (1.2171)
10 MB	36.6155 (0.4162)	36.1999 (0.6963)	36.3033 (0.4346)	34.3883 (0.4835)	21.1685 (5.6448)
1 MB	27.8291 (0.5119)	27.8370 (0.8441)	27.4086 (0.5610)	25.8917 (0.9571)	17.6334 (4.9234)
100 KB	7.4946 (0.1947)	7.5521 (0.2179)	7.5347 (0.1857)	7.4235 (0.2357)	6.3495 (1.3677)

[a]Numbers in parentheses are standard deviations.

[b]Sample session: Receiver: scps_ttcp -b 5900000 -G 0 -F 2 -R 80000000 -f m -r -s
Sender: scps_ttcp -b 5900000 -l 1024 -G 0 -F 2 -R 80000000 -f m -t destination < input file

Appendix G
SCPS-Pure Rate With Strictly Delayed Acknowledgments Single-Flow Test Results

The following table presents the average throughput and standard deviation for the SCPS-pure rate with strictly delayed acknowledgments single-flow test runs:

File size	Average throughput, Mbps[a]				
	Bit error rate, BER				
	0	10^{-8}	10^{-7}	10^{-6}	10^{-5}
Delay, 500 ms[b] [Buffer size, 5.7 MB; rate, 80 Mbps]					
100 MB	40.6889 (0.2610)	38.5426 (0.6644)	32.4833 (0.5477)	26.4906 (0.8445)	11.4047 (0.3863)
10 MB	27.4230 (0.1417)	26.3823 (1.1747)	22.6401 (1.5042)	18.8779 (1.4371)	8.9422 (1.0219)
1 MB	6.6064 (0.0192)	6.2164 (0.7313)	5.3497 (0.8814)	4.6762 (0.2783)	2.4753 (0.5666)
100 KB	.7778 (0.0002)	.7783 (0.0037)	.7519 (0.0768)	.6186 (0.1237)	.3887 (0.1585)
Delay, 250 ms [Buffer size, 3.1 MB; rate, 80 Mbps]					
100 MB	42.5819 (0.5334)	41.0768 (0.5488)	35.2755 (0.5295)	29.2944 (0.6444)	14.8907 (0.4447)
10 MB	33.6489 (0.6704)	32.7830 (1.0784)	28.6513 (1.2447)	23.4973 (1.6326)	12.9116 (2.3672)
1 MB	11.2542 (0.2322)	11.1645 (0.5050)	10.4362 (1.1536)	8.2890 (0.9201)	4.2911 (1.5495)
100 KB	1.5153 (0.0008)	1.5152 (0.0020)	1.4830 (0.1207)	1.0479 (0.2407)	.6825 (0.2953)
Delay, 10 ms [Buffer size, 2 MB; rate, 80 Mbps]					
100 MB	43.6973 (0.1921)	43.4323 (0.2403)	42.6451 (0.3605)	38.7626 (0.6586)	30.6357 (1.9131)
10 MB	42.6684 (0.5850)	42.8241 (0.2701)	42.0584 (0.3882)	37.9981 (0.5795)	29.2800 (5.3740)
1 MB	36.2813 (0.4302)	36.3149 (0.3003)	35.4964 (0.7942)	30.5330 (2.2978)	20.5624 (8.5477)
100 KB	7.3586 (0.2714)	7.4813 (0.2009)	7.4474 (0.2516)	7.4320 (0.2729)	5.7047 (2.0012)

[a]Numbers in parentheses are standard deviations.

[b]Sample session: Receiver: scps_ttcp -b 5700000 -G 0 -F 0 -R 80000000 -f m -r -s
Sender: scps_ttcp -b 5700000 -l 1024 -G 0 -F 0 -R 80000000 -f m -t destination < input file

Appendix H
MDP Single-Flow Test Results

The following table presents the average throughput and standard deviation for the MDP single-flow test runs:

(a) 500 ms delay[a]

File size, MB	Average throughput, Mbps[b,c]					
	Bit error rate, BER					
	0	10^{-8}	10^{-7}	10^{-6}	10^{-5}	10^{-4}
Rate, 20 Mbps						
100	DNC	18.6353 (0.0588)	18.6126 (0.2057)	18.2924 (0.2080)	DNC	DNC
10	18.5622 (0.0550)	18.2849 (0.9213)	15.3644 (2.3707)	12.9932 (0.7228)	9.0512 (1.1443)	1.6485 (0.7599)
1	3.1948 (0.0289)	3.6193 (1.5317)	3.4784 (1.3667)	2.7510 (0.5926)	1.5260 (0.3131)	0.2911 (0.0514)
Rate, 25 Mbps						
100	23.0091 (0.2424)	23.0211 (0.2086)	22.9347 (0.1086)	22.5331 (0.2938)	DNC	DNC
10	22.4796 (1.6314)	22.3412 (1.6661)	18.7406 (3.3108)	14.9246 (0.6748)	9.7262 (1.3342)	1.5257 (0.6886)
1	3.1785 (0.0404)	3.3141 (0.8943)	3.1735 (0.0434)	2.7386 (0.6249)	1.6019 (0.3406)	0.3105 (0.0548)
Rate, 30 Mbps						
100	27.1853 (0.1521)	27.1527 (0.1506)	27.0886 (0.3120)	26.4843 (0.7540)	DNC	DNC
10	27.1144 (0.1699)	26.6930 (1.0008)	20.6385 (4.2928)	16.8062 (1.7687)	9.6993 (0.9358)	1.4359 (0.5356)
1	3.8333 (1.2160)	3.3495 (1.0856)	3.0175 (0.3525)	2.5061 (0.5706)	1.6447 (0.3137)	0.2918 (0.0506)
Rate, 35 Mbps						
100	31.3851 (0.5424)	3.32404 (0.2126)	31.1757 (0.1598)	30.1167 (0.8586)	DNC	DNC
10	30.5169 (2.2655)	29.6565 (3.4740)	24.1353 (4.6950)	17.4630 (0.7770)	10.8349 (1.6101)	1.3826 (0.5531)
1	3.4240 (0.3407)	3.32404 (0.3704)	3.1917 (0.5153)	2.5690 (0.5635)	1.4184 (0.2293)	0.2812 (0.0421)
Rate, 40 Mbps						
100	35.2534 (0.6933)	35.0008 (0.2546)	34.8641 (0.8940)	33.0736 (1.0791)	20.9706 (1.2338)	DNC
10	33.4008 (4.8456)	31.3867 (7.1554)	22.0621 (3.9240)	19.9317 (1.9995)	9.7179 (1.4965)	1.3720 (0.5440)
1	3.6485 (0.7584)	3.7381 (0.6897)	2.7977 (0.8449)	2.2103 (0.1714)	1.3069 (0.2008)	0.2926 (0.0511)
Rate, 45 Mbps						
100	DNC	DNC	DNC	DNC	DNC	DNC
10	13.2007 (1.9086)	12.7810 (1.8853)	12.4146 (1.8894)	13.0448 (2.0159)	9.7625 (1.4087)	1.3568 (0.4930)
1	2.2289 (0.1882)	2.1919 (0.2363)	1.9269 (0.3611)	1.8744 (0.3532)	1.3190 (0.2366)	0.2906 (0.0467)
Rate, 50 Mbps						
100	DNC	DNC	DNC	DNC	DNC	DNC
10	7.7956 (0.5564)	7.6860 (0.6155)	7.9992 (0.9468)	7.6162 (0.8719)	7.0329 (0.9221)	1.3147 (0.4605)
1	1.5260 (0.1364)	1.5633 (0.0828)	1.5433 (0.1152)	1.4724 (0.1827)	1.2021 (0.2441)	DNC
Rate, 55 Mbps						
100	DNC	DNC	DNC	DNC	DNC	DNC
10	6.3110 (0.4238)	6.0691 (0.5867)	6.2278 (0.4006)	6.2451 (0.4748)	5.6666 (0.6820)	0.9841 (0.1076)
1	1.3024 (0.1657)	1.2775 (0.1605)	1.1944 (0.0522)	1.2340 (0.1291)	1.1162 (0.1198)	DNC
Rate, 60 Mbps						
100	DNC	DNC	DNC	DNC	DNC	DNC
10	5.1459 (0.5123)	5.1449 (0.3829)	5.2609 (0.2546)	5.0728 (0.4172)	5.0517 (0.4570)	0.9723 (0.1355)
1	1.2051 (0.0369)	1.2104 (0.0016)	1.2098 (0.0009)	1.1757 (0.0846)	1.0860 (0.1527)	DNC
Rate, 65 Mbps						
100	DNC	DNC	DNC	DNC	DNC	DNC
10	4.7177 (0.3652)	4.5340 (0.3193)	4.6952 (0.2908)	4.5977 (0.3057)	4.2549 (0.3518)	0.9252 (0.1158)
1	1.2112 (0.0013)	1.1868 (0.0720)	1.1752 (0.0853)	1.1632 (0.0956)	1.0350 (0.1382)	DNC
Rate, 70 Mbps						
100	DNC	DNC	DNC	DNC	DNC	DNC
10	4.4182 (0.2312)	4.2562 (0.3255)	4.4384 (0.2767)	4.6010 (0.3042)	3.8774 (0.3803)	0.9198 (0.1186)
1	1.1996 (0.0524)	1.1410 (0.1096)	1.1746 (0.0856)	1.0828 (0.1226)	0.9875 (0.1319)	DNC

[a]Sample session: Receiver: mdp -A (server address/port) -r (tx rate) -p 0 -X none -D (archive directory) -L (logfile)
Sender: mdp (client address/port) -r (tx rate) -p 0 -c -R 0 -1 1.0 (file or directory to transfer)

[b]DNC: "did not complete;" indicates the transfers were attempted, but they did not finish.

[c]Numbers in parentheses are standard deviations.

(a) Continued.—500 ms delay[a]

File size, MB	Average throughput, Mbps[b,c]					
	Bit error rate, BER					
	0	10^{-8}	10^{-7}	10^{-6}	10^{-5}	10^{-4}
Rate, 75 Mbps						
100	DNC	DNC	DNC	DNC	DNC	DNC
10	4.7872 (0.2863)	4.6004 (0.4178)	4.5406 (0.4186)	4.4637 (0.3172)	4.1164 (0.4280)	0.9243 (0.1141)
1	1.1992 (0.0517)	1.1869 (0.0720)	1.1977 (0.0521)	1.1534 (0.1033)	1.0445 (0.1489)	DNC
Rate, 80 Mbps						
100	DNC	DNC	DNC	DNC	DNC	DNC
10	4.6617 (0.2786)	4.4682 (0.4691)	4.5657 (0.2975)	4.6308 (0.3319)	4.2568 (0.3963)	0.8536 (0.1251)
1	1.1881 (0.0718)	1.2101 (0.0018)	1.1753 (0.0849)	1.1560 (0.0984)	1.0243 (0.1223)	DNC

[a]Sample session: Receiver: mdp -A (server address/port) -r (tx rate) -p 0 -X none -D (archive directory) -L (logfile)
Sender: mdp (client address/port) -r (tx rate) -p 0 -c -R 0 -1 1.0 (file or directory to transfer)

[b]DNC: "did not complete;" indicates the transfers were attempted, but they did not finish.

[c]Numbers in parentheses are standard deviations.

(b) 250 ms delay[a]

File size, MB	Average throughput, Mbps[b,c]					
	Bit error rate, BER					
	0	10^{-8}	10^{-7}	10^{-6}	10^{-5}	10^{-4}
Rate, 20 Mbps						
100	DNC	18.7749 (0.0355)	18.7255 (0.0735)	18.5663 (0.0092)	16.5047 (0.2113)	1.9356 (0.0766)
10	18.5633 (0.0549)	18.4142 (0.7548)	16.6425 (2.1611)	13.9632 (0.6018)	9.6646 (1.0547)	1.4156 (0.6321)
1	3.9715 (0.0565)	4.1384 (1.2669)	3.9258 (0.2497)	3.3742 (0.7716)	1.9302 (0.3648)	0.3264 (0.0500)
Rate, 25 Mbps						
100	23.2811 (0.2573)	23.2158 (0.0514)	23.1941 (0.0493)	22.8348 (0.3072)	19.5179 (1.0538)	1.8546 (0.0436)
10	22.8188 (0.2514)	22.8525 (0.1678)	19.9806 (3.1282)	16.0166 (2.0483)	10.2206 (1.2646)	1.4223 (0.5548)
1	3.9767 (0.0576)	4.1581 (1.2387)	3.9338 (0.2523)	3.5305 (0.7194)	1.9155 (0.3821)	0.3255 (0.0507)
Rate, 30 Mbps						
100	27.5201 (0.0972)	27.5618 (0.0436)	27.5219 (0.3642)	26.8261 (0.6391)	22.8524 (1.1234)	14.2666 (17.5879)
10	27.0979 (0.1785)	26.8974 (0.3897)	20.8003 (3.6668)	17.9618 (1.5815)	10.7191 (1.5222)	1.4125 (0.5281)
1	3.9637 (0.0690)	3.9585 (0.0748)	4.3727 (1.7209)	3.4279 (0.7292)	1.9789 (0.3904)	0.3363 (0.0587)
Rate, 35 Mbps						
100	31.7579 (0.0410)	31.6653 (0.1250)	31.6185 (0.3501)	30.4605 (0.8714)	22.9759 (1.7208)	DNC
10	30.8724 (1.8723)	29.9796 (2.8974)	26.2167 (4.8136)	17.8453 (4.3060)	11.7108 (1.9650)	1.4442 (0.5632)
1	4.1879 (0.2360)	4.1029 (0.4220)	3.7597 (0.6887)	3.1185 (0.7839)	1.7991 (0.3646)	.3449 (0.0521)
Rate, 40 Mbps						
100	35.6611 (0.1301)	35.6811 (0.1376)	35.3800 (0.8578)	34.0463 (0.8692)	DNC	DNC
10	32.3341 (5.6975)	32.7924 (4.9799)	22.2251 (0.1839)	20.9990 (2.2823)	11.1985 (1.3022)	1.4675 (0.5268)
1	4.2228 (0.8125)	3.7343 (1.0125)	3.4067 (1.0252)	2.5488 (0.2412)	1.5414 (0.2569)	.3372 (0.0512)
Rate, 45 Mbps						
100	DNC	DNC	DNC	DNC	DNC	DNC
10	15.3489 (1.8885)	15.5814 (1.7851)	15.8119 (1.6157)	14.3859 (1.9849)	11.1555 (1.8366)	1.4158 (0.5771)
1	2.6412 (0.1286)	2.6737 (0.2626)	2.5363 (0.2930)	2.2729 (0.4096)	1.6556 (0.2480)	.3167 (0.0634)
Rate, 50 Mbps						
100	DNC	DNC	DNC	DNC	DNC	DNC
10	9.3556 (0.9494)	9.3576 (0.6232)	9.5304 (0.7574)	9.2673 (0.6216)	8.1506 (1.1590)	1.4202 (0.5448)
1	1.8548 (0.0182)	1.8573 (0.0024)	1.8324 (0.0976)	1.8020 (0.1454)	1.4615 (0.2495)	.3362 (0.0699)
Rate, 55 Mbps						
100	DNC	DNC	DNC	DNC	DNC	DNC
10	7.5192 (1.1260)	7.5754 (0.9857)	7.0083 (1.1006)	7.2084 (0.4915)	6.2211 (0.5777)	1.0380 (0.1510)
1	1.4779 (0.1478)	1.5548 (0.2045)	1.4887 (0.1587)	1.4460 (0.1579)	1.3392 (0.1450)	.3290 (0.0535)
Rate, 60 Mbps						
100	DNC	DNC	DNC	DNC	DNC	DNC
10	6.3265 (0.4138)	6.5990 (0.3931)	6.6182 (0.3483)	6.5032 (0.3524)	5.8141 (1.0647)	1.0102 (0.1190)
1	1.4202 (0.0435)	1.4181 (0.0448)	1.4263 (0.0019)	1.3857 (0.0984)	1.2484 (0.1463)	.3148 (0.0665)
Rate, 65 Mbps						
100	DNC	DNC	DNC	DNC	DNC	DNC
10	6.1680 (0.3733)	6.4998 (0.4911)	6.3225 (0.3067)	6.3956 (0.3959)	5.7024 (0.7780)	1.0856 (0.1404)
1	1.4001 (0.0834)	1.3684 (0.1176)	1.4253 (0.0016)	1.3982 (0.0832)	1.2324 (0.1806)	DNC
Rate, 70 Mbps						
100	DNC	DNC	DNC	DNC	DNC	DNC
10	6.1817 (0.3962)	6.2957 (0.2843)	6.1592 (0.2933)	6.1967 (0.3228)	5.5106 (0.6303)	0.9969 (0.1380)
1	1.3864 (0.1002)	1.4123 (0.0616)	1.4013 (0.0777)	1.3452 (0.1199)	1.2144 (0.1977)	DNC
Rate, 75 Mbps						
100	DNC	DNC	DNC	DNC	DNC	DNC
10	6.0576 (0.3623)	6.0533 (0.5351)	6.0574 (0.3869)	6.0006 (0.4390)	5.5759 (0.5664)	1.0907 (0.1339)
1	1.4268 (0.0024)	1.3585 (0.1211)	1.3996 (0.0839)	1.3712 (0.1111)	1.2326 (0.1574)	DNC
Rate, 80 Mbps						
100	DNC	DNC	DNC	DNC	DNC	DNC
10	5.8169 (0.4053)	5.7169 (0.3406)	5.7999 (0.3751)	5.5870 (0.5791)	5.3535 (0.6235)	1.0179 (0.1229)
1	1.3720 (0.1120)	1.3586 (0.1207)	1.3976 (0.0838)	1.3984 (0.0833)	1.1709 (0.1559)	DNC

[a]Sample session: Receiver: mdp -A (server address/port) -r (tx rate) -p 0 -X none -D (archive directory) -L (logfile)
Sender: mdp (client address/port) -r (tx rate) -p 0 -c -R 0 -1 1.0 (file or directory to transfer)

[b]DNC: "did not complete;" indicates the transfers were attempted, but they did not finish.

[c]Numbers in parentheses are standard deviations.

(c) 10 ms delay[a]

File size, MB	Average throughput, Mbps[b,c]					
	Bit error rate, BER					
	0	10^{-8}	10^{-7}	10^{-6}	10^{-5}	10^{-4}
Rate, 20 Mbps						
100	DNC	18.8784 (0.0412)	18.8811 (0.0092)	18.7274 (0.0283)	16.8083 (0.1792)	2.8926 (0.1025)
10	18.5664 (0.0523)	18.4571 (0.5857)	16.6257 (1.9774)	14.4102 (0.8125)	10.3181 (1.0037)	1.6166 (0.6653)
1	4.6916 (0.0359)	4.6442 (0.2893)	4.6041 (0.4040)	4.2413 (0.7998)	2.2434 (0.4195)	0.3596 (0.0792)
Rate, 25 Mbps						
100	23.4112 (0.0123)	23.3876 (0.0292)	23.3075 (0.2214)	23.0439 (0.2969)	19.6639 (1.4646)	2.7528 (0.1512)
10	22.8327 (0.3056)	22.8984 (0.1014)	20.5839 (2.7141)	17.1139 (0.8754)	11.5568 (1.2620)	1.4675 (0.4789)
1	4.9138 (0.0405)	4.9178 (0.0443)	5.1238 (1.6159)	4.1694 (0.9560)	2.2507 (0.4600)	0.4064 (0.0826)
Rate, 30 Mbps						
100	27.7703 (0.0774)	27.7911 (0.0388)	27.7706 (0.0249)	27.0454 (0.5022)	22.8985 (1.2811)	2.7299 (0.0663)
10	26.7226 (1.6246)	27.0115 (0.4227)	21.6327 (3.1605)	18.9693 (1.6045)	12.3589 (1.5545)	1.3972 (0.4524)
1	5.0752 (0.0598)	5.0179 (0.3277)	4.9763 (0.4598)	4.4531 (0.9546)	2.4124 (0.5134)	0.4110 (0.0933)
Rate, 35 Mbps						
100	32.0643 (0.0320)	32.0595 (0.0517)	32.0993 (0.3344)	31.1948 (0.6632)	23.3506 (2.2669)	2.7233 (0.0702)
10	30.9421 (1.5186)	30.8528 (1.4545)	28.3160 (4.2708)	21.4997 (1.0967)	13.2693 (1.5929)	1.4487 (0.4277)
1	5.1178 (0.4832)	5.2412 (0.0350)	4.5793 (1.0107)	3.7450 (1.0610)	2.4059 (0.6424)	0.3770 (0.0689)
Rate, 40 Mbps						
100	36.1704 (0.0566)	36.0559 (0.1974)	36.0586 (0.1263)	34.7050 (0.7644)	22.4467 (1.3537)	2.7278 (0.0625)
10	32.7360 (5.1617)	31.9727 (5.1453)	24.6460 (2.8029)	21.9529 (2.5904)	12.2394 (1.7810)	1.5469 (0.7385)
1	5.0488 (0.820)	4.1987 (1.1474)	3.8017 (1.0524)	3.0974 (0.1462)	1.9736 (0.6050)	0.3911 (0.0408)
Rate, 45 Mbps						
100	DNC	DNC	DNC	DNC	DNC	DNC
10	16.8193 (2.2182)	17.3048 (1.6999)	17.3694 (1.7037)	17.1158 (1.8469)	12.7991 (1.7961)	1.5569 (0.6558)
1	3.1951 (0.3663)	3.1419 (0.1491)	3.1145 (0.2111)	2.8643 (0.4638)	2.2669 (0.6378)	0.3679 (0.0627)
Rate, 50 Mbps						
100	DNC	DNC	DNC	DNC	DNC	DNC
10	10.5793 (0.6287)	10.6282 (0.4936)	10.8079 (0.6978)	10.1884 (0.8785)	9.5043 (0.8794)	1.5261 (0.4598)
1	2.2316 (0.0038)	2.5878 (0.5085)	2.6479 (0.4985)	2.4854 (0.5462)	1.8511 (0.5700)	0.3930 (0.0906)
Rate, 55 Mbps						
100	DNC	DNC	DNC	DNC	DNC	DNC
10	8.5899 (0.7396)	8.6728 (0.6211)	8.2592 (0.4147)	8.6036 (0.9377)	7.4527 (0.6627)	1.1312 (0.1721)
1	2.0014 (0.2619)	1.9472 (0.3784)	1.9321 (0.2867)	1.7370 (0.1601)	1.5772 (0.1875)	0.3917 (0.0752)
Rate, 60 Mbps						
100	DNC	DNC	DNC	DNC	DNC	DNC
10	7.7087 (0.4837)	7.7620 (0.5653)	7.6376 (0.4141)	7.4328 (0.4955)	6.7433 (0.6395)	1.1826 (0.1610)
1	1.7054 (0.0712)	1.7121 (0.0530)	1.7468 (0.2263)	1.6647 (0.1249)	1.5126 (0.1676)	0.4011 (0.0931)
Rate, 65 Mbps						
100	DNC	DNC	DNC	DNC	DNC	DNC
10	6.4950 (1.5725)	7.5674 (0.5697)	7.7746 (0.4279)	7.3914 (0.4047)	6.6029 (0.6269)	1.1582 (0.1268)
1	1.7053 (0.0734)	1.6867 (0.0988)	1.7402 (0.3500)	1.6376 (0.1430)	1.4083 (0.1934)	DNC
Rate, 70 Mbps						
100	DNC	DNC	DNC	DNC	DNC	DNC
10	7.3603 (0.4281)	7.4662 (0.3087)	7.5642 (0.4565)	7.2213 (0.5007)	6.6450 (0.7169)	1.1502 (0.1883)
1	1.6898 (0.1010)	1.6563 (0.1328)	1.6524 (0.1326)	1.6858 (0.1004)	1.3980 (0.1449)	DNC
Rate, 75 Mbps						
100	DNC	DNC	DNC	DNC	DNC	DNC
10	7.3852 (0.2814)	7.5215 (0.4942)	7.2972 (0.3332)	7.3445 (0.7720)	6.4144 (0.7495)	1.0737 (0.1650)
1	1.7060 (0.0722)	1.1814 (0.2774)	1.5874 (0.1614)	1.6361 (0.1431)	1.4070 (0.1877)	DNC
Rate, 80 Mbps						
100	DNC	DNC	DNC	DNC	DNC	DNC
10	7.3587 (0.3496)	7.4452 (0.443)	7.2414 (0.5257)	7.1310 (0.3269)	6.3227 (0.6970)	1.1779 (0.1696)
1	1.7065 (0.0724)	0.8775 (0.148)	1.6370 (0.1433)	1.6220 (0.1539)	DNC	DNC

[a]Sample session: Receiver: mdp -A (server address/port) -r (tx rate) -p 0 -X none -D (archive directory) -L (logfile)
Sender: mdp (client address/port) -r (tx rate) -p 0 -c -R 0 -1 1.0 (file or directory to transfer)

[b]DNC: "did not complete;" indicates the transfers were attempted, but they did not finish.

[c]Numbers in parentheses are standard deviations.

(d) 0 ms delay

File size, MB	Average throughput, Mbps[b,c]					
	Bit error rate, BER					
	0	10^{-8}	10^{-7}	10^{-6}	10^{-5}	10^{-4}
Rate, 20 Mbps						
100	DNC	18.8988 (0.0212)	18.9355 (0.2771)	18.7429 (0.0093)	16.5853 (0.3509)	2.9362 (0.0557)
10	18.5316 (0.1448)	18.5500 (0.0516)	16.8720 (1.8496)	14.7666 (0.0365)	10.4842 (1.1964)	1.5015 (0.5501)
1	4.7139 (0.0365)	4.9321 (1.2993)	4.5389 (0.5567)	4.0846 (0.8808)	2.2079 (0.5024)	0.4069 (0.1013)
Rate, 25 Mbps						
100	23.4242 (0.0144)	23.4389 (0.1660)	23.4298 (0.1750)	23.1332 (0.2321)	20.2435 (0.5022)	2.8771 (0.0891)
10	22.8814 (0.1652)	22.3631 (1.6331)	20.0404 (2.7369)	17.0579 (1.0009)	11.8914 (1.0533)	1.6001 (0.6315)
1	5.1786 (1.4831)	5.1895 (1.5739)	4.9036 (0.3123)	4.3378 (0.9111)	2.2925 (0.4641)	0.3958 (0.0718)
Rate, 30 Mbps						
100	27.8344 (0.0051)	27.7987 (0.0712)	27.7347 (0.2460)	27.1816 (0.6645)	22.8396 (1.0383)	2.7935 (0.0981)
10	27.1274 (0.1609)	26.7657 (1.6079)	21.8532 (3.2763)	19.0397 (1.7079)	12.5127 (1.3406)	1.5285 (0.6108)
1	5.1089 (0.0589)	5.1085 (0.0546)	4.9457 (0.5593)	4.6430 (0.8806)	2.4185 (0.5769)	0.4042 (0.0706)
Rate, 35 Mbps						
100	32.0460 (0.0691)	32.0572 (0.0705)	31.9527 (0.2958)	30.9170 (0.6283)	23.7044 (1.8713)	2.7732 (0.0984)
10	31.2039 (0.2442)	30.6865 (2.0891)	28.6200 (4.1094)	21.7173 (0.8077)	13.3300 (1.8952)	1.5182 (0.5628)
1	5.2561 (0.0404)	5.1624 (0.4852)	4.9974 (0.7368)	4.3447 (1.0981)	2.4810 (0.6280)	0.4056 (0.0612)
Rate, 40 Mbps						
100	36.1724 (0.0837)	36.0963 (0.1999)	35.8504 (0.6408)	34.6881 (1.0124)	22.6923 (0.8863)	2.7791 (0.1147)
10	33.4041 (4.0840)	32.9105 (4.6281)	24.4674 (3.2786)	22.1185 (2.8296)	12.3609 (1.6942)	1.5728 (0.5389)
1	5.0257 (0.8789)	4.5150 (1.1328)	3.9446 (1.1018)	3.0999 (0.2056)	2.1955 (0.6279)	0.3820 (0.0707)
Rate, 45 Mbps						
100	DNC	DNC	DNC	DNC	DNC	DNC
10	16.9049 (2.3995)	16.5244 (2.1165)	16.9304 (2.4054)	16.0905 (2.4034)	12.5706 (1.7906)	1.5270 (0.5251)
1	3.1469 (0.2110)	3.1886 (0.0098)	3.1124 (0.2538)	2.9947 (0.3847)	2.0204 (0.5537)	0.3850 (0.0760)
Rate, 50 Mbps						
100	DNC	DNC	DNC	DNC	DNC	DNC
10	10.8958 (0.7103)	10.8288 (0.716)	11.3430 (1.7251)	10.6999 (0.9447)	9.5633 (1.094)	1.6369 (0.5905)
1	2.2388 (0.0823)	2.5264 (0.465)	2.5833 (0.5415)	2.3710 (0.5680)	1.8659 (0.487)	0.3760 (0.0817)
Rate, 55 Mbps						
100	DNC	DNC	DNC	DNC	DNC	DNC
10	8.5518 (0.8603)	8.7781 (0.8978)	8.6865 (0.8977)	8.2420 (0.6988)	7.2269 (1.1618)	1.0626 (0.1494)
1	2.0053 (0.2664)	1.9648 (0.3296)	1.9269 (0.2594)	1.6889 (0.1092)	1.5923 (0.2258)	0.3995 (0.0782)
Rate, 60 Mbps						
100	DNC	DNC	DNC	DNC	DNC	DNC
10	7.7422 (0.4247)	7.7315 (0.4296)	7.9471 (0.6451)	7.6274 (0.4132)	7.2620 (1.1381)	1.1093 (0.1249)
1	1.7290 (0.0521)	1.6457 (0.1489)	1.6857 (0.1187)	1.6780 (0.2798)	1.4625 (0.2029)	0.4292 (0.0833)
Rate, 65 Mbps						
100	DNC	DNC	DNC	DNC	DNC	DNC
10	7.3073 (0.5006)	7.6594 (0.4876)	8.5300 (1.7801)	7.3293 (0.6013)	6.4658 (0.6044)	1.1466 (0.1441)
1	1.7207 (0.0729)	1.5363 (0.1640)	1.5849 (0.1658)	1.5509 (0.1652)	1.4246 (0.1642)	DNC
Rate, 70 Mbps						
100	DNC	DNC	DNC	DNC	DNC	DNC
10	7.5358 (0.4688)	7.5184 (0.3655)	7.5395 (0.5747)	7.4471 (0.3000)	6.9122 (0.4621)	1.2079 (0.2456)
1	1.7372 (0.0034)	1.5702 (0.1674)	1.5872 (0.1661)	1.6350 (0.1524)	1.3396 (0.1615)	DNC
Rate, 75 Mbps						
100	DNC	DNC	DNC	DNC	DNC	DNC
10	7.5093 (0.5255)	7.7222 (1.7532)	7.9039 (0.4974)	7.2821 (0.5989)	6.3919 (0.6325)	1.1633 (0.1447)
1	1.7207 (0.0738)	1.5379 (0.1654)	1.6197 (0.1603)	1.7370 (0.3529)	1.3603 (0.2018)	DNC
Rate, 80 Mbps						
100	DNC	DNC	DNC	DNC	DNC	DNC
10	7.2561 (0.3017)	7.3280 (0.4717)	7.7065 (0.5210)	7.3721 (0.4574)	6.3608 (0.5748)	1.0940 (0.1243)
1	1.7372 (0.0042)	1.6952 (0.5238)	1.5425 (0.1837)	1.5532 (0.1668)	1.3383 (0.1546)	DNC

[b]DNC: "did not complete;" indicates the transfers were attempted, but they did not finish.
[c]Numbers in parentheses are standard deviations.

Appendix I
MFTP Single-Flow Test Results

The following table presents the average throughput and standard deviation for the MFTP single-flow test runs:[a]

File size	Average throughput, Mbps[b]				
	Bit error rate, BER				
	0	10^{-8}	10^{-7}	10^{-6}	10^{-5}
Delay, 500 ms					
100 MB	18.9987 (0.5367)	18.5346 (0.6938)	18.3568 (0.4921)	14.867 (0.3381)	7.6358 (0.1495)
10 MB	14.5358 (1.2462)	13.7696 (1.3703)	12.6157 (0.2382)	9.8574 (0.8240)	5.8939 (0.2785)
1 MB	3.4292 (0.0136)	3.3240 (0.3229)	2.7537 (0.5279)	2.3216 (0.1393)	1.4005 (0.2401)
100 KB	.2590 (0.0736)	.2806 (0.0706)	.2724 (0.0954)	.2199 (0.0717)	.1599 (0.0540)
Delay, 250 ms					
100 MB	18.9792 (0.4570)	18.6650 (0.5765)	18.3619 (0.2694)	14.5662 (0.5064)	7.8929 (0.1738)
10 MB	14.4617 (1.2176)	14.0097 (1.3548)	12.7070 (0.1085)	10.1158 (0.8655)	6.3526 (0.4884)
1 MB	3.4343 (0.0122)	3.3918 (0.1963)	3.0760 (0.5158)	2.2848 (0.1942)	1.5197 (0.2083)
100 KB	.3118 (0.0859)	.3206 (0.0877)	.3346 (0.1055)	.2457 (0.0882)	.1830 (0.0633)
Delay, 10 ms					
100 MB	19.1125 (0.7107)	18.9014 (0.9808)	18.7379 (0.5016)	15.0342 (0.6009)	7.8991 (0.1643)
10 MB	14.7334 (1.0835)	14.0473 (1.3550)	12.7335 (0.0336)	10.8161 (1.2054)	6.5070 (0.6003)
1 MB	3.4176 (0.0603)	3.3221 (0.3274)	2.6640 (0.5183)	2.3176 (0.1429)	1.5746 (0.1939)
100 KB	.3784 (0.0758)	.3540 (0.0888)	.3606 (0.0842)	.2413 (0.0834)	.1774 (0.0684)
Delay, 0 ms					
100 MB	19.1953 (0.3154)	19.5706 (0.8321)	19.4823 (0.9250)	15.9199 (0.4798)	7.8290 (2.4386)
10 MB	15.2130 (0.2513)	13.5910 (1.4179)	12.5663 (0.5843)	10.6042 (0.5471)	6.4800 (0.5467)
1 MB	3.4292 (0.0136)	3.2822 (0.3774)	2.6742 (0.4913)	2.2856 (0.1954)	1.4685 (0.1904)
100 KB	.3829 (0.0729)	.3518 (0.0866)	.3451 (0.0904)	.2695 (0.0933)	.1870 (0.0607)

[a]Sample session: Receiver: sbclient_client -l (logfile)
Sender: sbserver_cli -c (config file) -g (group file) -l (logfile) -m FILE

[b]Numbers in parentheses are standard deviations.

Appendix J
Test Bed System Information

Information on the hardware and software used for all of the tests is provided in the following:

SENDING MACHINES

Processor	Operating system	CPU speed, MHz	Random access memory, RAM, MB
Sun Ultra II[a]	Solaris 7	200	512
Sun Ultra I	Solaris 7	143	64
Sun Ultra 10	Solaris 8	440	132

RECEIVING MACHINES

Processor	Operating system	CPU speed, MHz	Random access memory, RAM, MB
Sun Ultra II[a]	Solaris 7	200	256
Sun Ultra 5	Solaris 7	270	256
Sun Ultra 10	Solaris 8	440	132

MONITORING (TRACING) MACHINES—SENDER

Processor	Operating system	CPU speed, MHz	Random access memory, RAM, MB
Pentium III[a]	NetBSD	550	64
Pentium II	NetBSD	450	400
Pentium Pro	NetBSD	200	64

MONITORING (TRACING) MACHINES—RECEIVER

Processor	Operating system	CPU speed, MHz	Random access memory, RAM, MB
Pentium III[a]	NetBSD	550	64
Pentium II	NetBSD	450	400
Mac PPC 604	NetBSD	120	64

[a]These rows designate equipment used in single-flow tests. All networking equipment were used in both the single- and multiple-flow tests.

Additional networking equipment:

- Adtech SX/14 channel simulator
- Cisco Catalyst 2900 Ethernet switch (two)
- Cisco 7100 router (two)

CISCO ROUTER MEMORY

	Memory, bytes				
	Total	Used	Free	Lowest	Largest
Terrestrial					
Processor	25 779 136	9 330 364	16 448 772	16 332 616	16 348 656
I/O	67 108 872	3 673 752	63 435 120	63 435 120	63 435 068
I/O-2	8 388 616	2 716 696	5 671 920	5 671 920	5 671 868
Space					
Processor	25 779 136	9 331 260	16 447 876	16 331 880	16 348 464
I/O	67 108 872	3 673 752	63 435 120	63 435 120	63 435 068
I/O-2	8 388 616	2 716 696	5 671 920	5 671 920	5 671 868

Cisco Router Buffers (Terrestrial)

- Buffer elements:
  499 in free list (500 max. allowed)
- Public Buffer pools:
  Small buffer, 104 bytes (total 256, permanent 256)
  250 in free list (64 min., 1280 max. allowed)
  Middle buffer, 600 bytes (total 256, permanent 256)
  254 in free list (64 min., 1280 max. allowed)
  Big buffer, 1524 bytes (total 256, permanent 256)
  256 in free list (64 min., 1280 max. allowed)
  Very big buffer, 4520 bytes (total 256, permanent 256)
  256 in free list (64 min., 1280 max. allowed)
  Large buffer, 5024 bytes (total 256, permanent 256)
  256 in free list (64 min., 1280 max. allowed)
  Huge buffer, 18024 bytes (total 16, permanent 16)
  16 in free list (4 min., 64 max. allowed)
- Interface buffer pools:
  IPC buffers, 4096 bytes
- Header pools:
  Header buffer, 0 bytes (total 511, permanent 256)
  255 in free list (256 min., 1024 max. allowed)
- Particle clones:
  1024 clones
- Public particle pools:
  F/S buffers, 128 bytes (total 512, permanent 512)
  0 in free list (0 min., 512 max. allowed)
  Normal buffers, 512 bytes (total 1024, permanent 1024)
  1024 in free list (512 min., 2048 max. allowed)
- Private particle pools:
  FastEthernet 0/0 buffer, 512 bytes (total 400, permanent 400)
  0 in free list (0 min., 400 max. allowed)
  FastEthernet 0/1 buffer, 512 bytes (total 400, permanent 400)
  0 in free list (0 min., 400 max. allowed)
  ATM 1/0 buffer, 512 bytes (total 1200, permanent 1200)
  0 in free list (0 min., 1200 max. allowed)
  ATM 2/0 buffer, 512 bytes (total 1200, permanent 1200)
  0 in free list (0 min., 1200 max. allowed)

Cisco Router Buffers (Space)

- Buffer elements:
  499 in free list (500 max. allowed)
- Public Buffer pools:
  Small buffer, 104 bytes (total 256, permanent 256)
    252 in free list (64 min., 1280 max. allowed)
  Middle buffer, 600 bytes (total 256, permanent 256)
    252 in free list (64 min., 1280 max. allowed)
  Big buffer, 1524 bytes (total 256, permanent 256)
    256 in free list (64 min., 1280 max. allowed)
  Very big buffer, 4520 bytes (total 256, permanent 256)
    256 in free list (64 min., 1280 max. allowed)
  Large buffer, 5024 bytes (total 256, permanent 256)
    256 in free list (64 min., 1280 max. allowed)
  Huge buffer, 18024 bytes (total 16, permanent 16)
    16 in free list (4 min., 64 max. allowed)
- Interface buffer pools:
  IPC buffers, 4096 bytes
- Header pools:
  Header buffer, 0 bytes (total 511, permanent 256)
    5 in free list (256 min., 1024 max. allowed)
- Particle clones:
  1024 clones
- Public particle pools:
  F/S buffers, 128 bytes (total 512, permanent 512)
    0 in free list (0 min., 512 max. allowed)
  Normal buffers, 512 bytes (total 1024, permanent 1024)
    1024 in free list (512 min., 2048 max. allowed)
- Private particle pools:
  FastEthernet 0/0 buffer, 512 bytes (total 400, permanent 400)
    0 in free list (0 min., 400 max. allowed)
  FastEthernet 0/1 buffer, 512 bytes (total 400, permanent 400)
    0 in free list (0 min., 400 max. allowed)
  ATM 1/0 buffer, 512 bytes (total 1200, permanent 1200)
    0 in free list (0 min., 1200 max. allowed)
  ATM 2/0 buffer, 512 bytes (total 1200, permanent 1200)
    0 in free list (0 min., 1200 max. allowed)

Appendix K
Theoretical Throughput of Congestion-Based Protocols

The following table presents the calculated throughput values of the congestion-based protocols:

Bit error rate, BER	Throughput, Mbps		
	Delay, ms		
	500	250	10
0	90.8540	86.3113	94.6396
10^{-8}	1.5630	3.1261	78.1518
10^{-7}	.4944	.9887	24.7185
10^{-6}	.1566	.3133	7.8316
10^{-5}	.0505	.1010	2.5241
10^{-4}	.0190	.0381	.9524

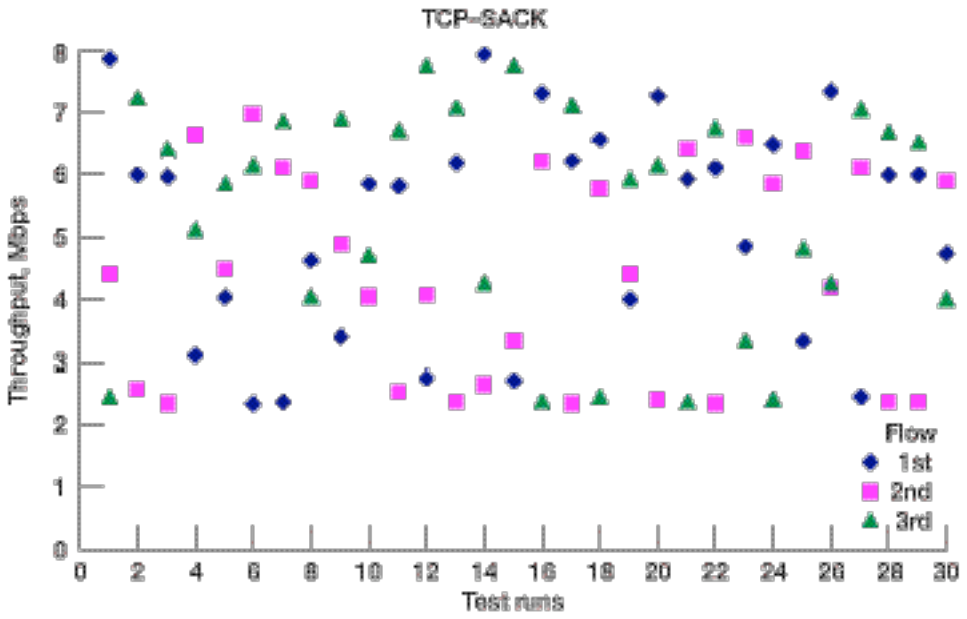
Appendix L
Theoretical Throughput of Rate-Based Protocols

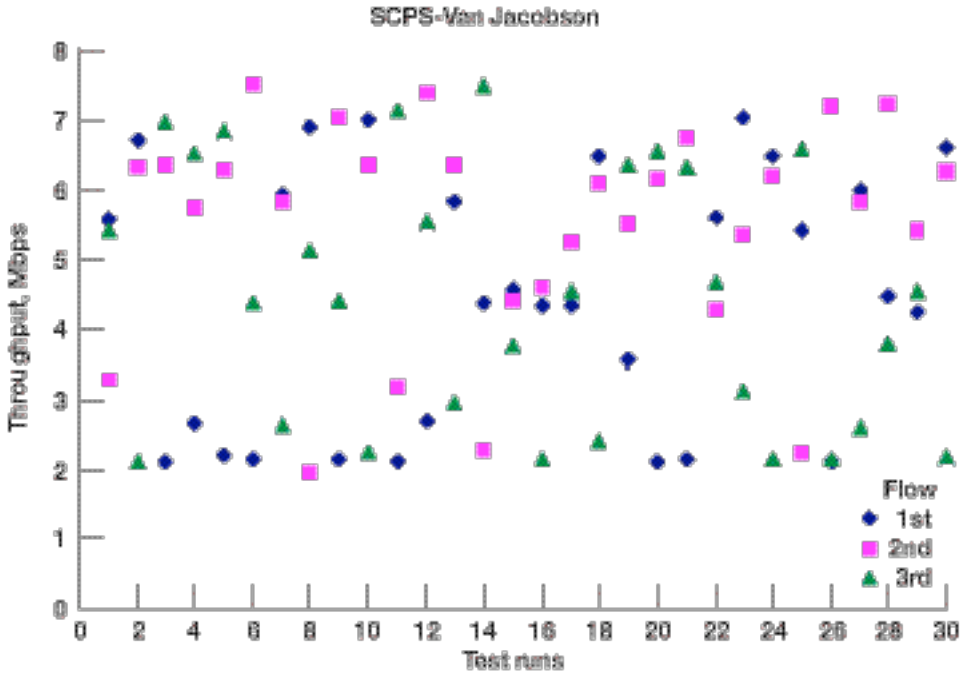
The following table presents the calculated throughput values of the rate-based protocols:

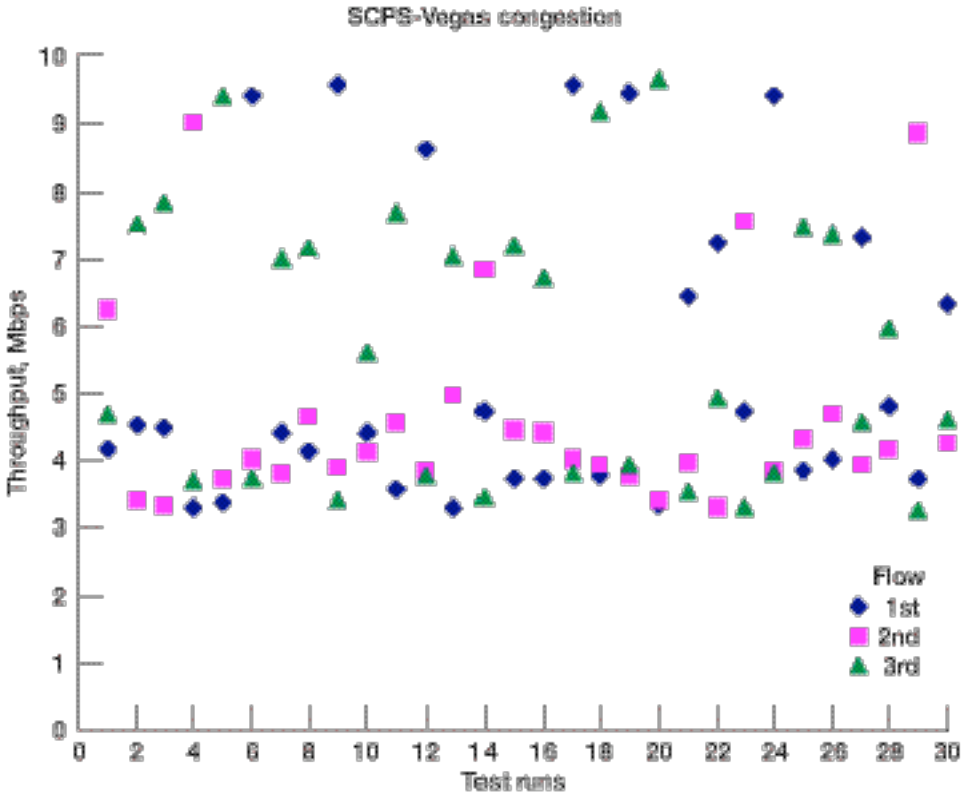
File size	Throughput, Mbps					
	Bit error rate, BER					
	0	10^{-8}	10^{-7}	10^{-6}	10^{-5}	10^{-4}
Delay, 500 ms						
100 MB	72.1063	72.1005	72.0479	71.5289	66.9068	46.6459
10 MB	50.4744	50.4716	50.4458	50.1908	47.8703	36.5207
1 MB	12.6186	12.6184	12.6168	12.6008	12.4493	11.5184
100 KB	1.4845	1.4845	1.4845	1.2686	1.4822	1.4680
Delay, 250 ms						
100 MB	73.8650	73.8589	73.8038	73.2592	68.4184	47.3756
10 MB	60.5693	60.5652	60.5281	60.1614	56.8577	41.5287
1 MB	21.6319	21.6314	21.6266	21.5796	21.1391	18.5881
100 KB	2.9120	2.9120	2.9119	2.9110	2.9029	2.8492
Delay, 10 ms						
100 MB	75.6360	75.6296	75.5718	75.0009	69.9351	48.0979
10 MB	74.9620	74.9557	74.8989	74.3382	69.3585	47.8245
1 MB	68.8288	68.8234	68.7756	68.3025	64.0756	45.2519
100 KB	37.8558	37.8542	37.8397	37.6961	36.3719	29.4241

Appendix M
Multiple-Flow Test Results

The following diagrams show examples of the throughputs from 30 test runs of the TCP–SACK, SCPS-Van Jacobson, and SCPS-Vegas congestion protocols in multiple-flow tests:






SCPS-Vegas congestion
Throughput, Mbps
Test runs
Flow
1st
2nd
3rd
0
1
2
3
4
5
6
7
8
9
10
0
2
4
6
8
10
12
14
16
18
20
22
24
26
28
30

References

1. Ernst, Darrell E.; and Durst, Robert C.: Space Communications Protocol Standards (SCPS) Bent-Pipe Experiment Report. SCPS–D71.51–Y–1, 1996.
2. Horan, S.; and Wang, R.: Comparison of File Transfer Using SCPS FP and TCP/IP FTP Over a Simulated Satellite Channel. Proceedings of Telemetering Conference, vol. 34, 1999, pp. 87–95.
3. Muhonen, John; and Durst, Robert C.: Space Communications Protocol Standards (SCPS) FY97 DOD Test Report. MTRE Technical Report MTR 98B0000011, 1998.
4. Edwards, Eric, et al.: Performance of SCPS in Complex Networks. AIAA Paper 2002–1912, 2002.
5. Jacobson, Van; and Karels, Michael J.: Congestion Avoidance and Control. Comput. Commun. Rev., vol. 18, no. 4, 1988.
6. Stevens, Richard W.: TCP/IP Illustrated. Vol. 1, chapt. 20, Addison-Wesley Publishing Co., Reading, MA, 1994.
7. Stevens, Richard W.: TCP/IP Illustrated. Vol. 1, chapt. 21, Addison-Wesley Publishing Co., Reading, MA, 1994.
8. Mathis, M., et al.: TCP Selective Acknowledgment Options. RFC 2018, 1996.
9. Durst, Robert C.; Miller, Gregory J.; and Travis, Eric J.: TCP Extensions for Space Communications. http://www.scps.org/Documents/TCPExt4Sat.pdf Accessed Jan. 7, 2003.
10. Fox, R.: TCP Big Window and NAK Options. RFC 1106, 1989. http://www.faqs.org/rfcs/rfc1106.html Accessed Jan. 7, 2003.
11. Brakmo, L.S.; O'Malley, S.W.; and Peterson, L.L.: TCP Vegas: New Techniques for Congestion Detection and Avoidance. Comput. Commun. Rev., vol. 24, no. 4, 1994.
12. Macker, Joe; and Dang, Winston: The Multicast Dissemination Protocol (MDP) Version 1 Framework. Naval Research Laboratory Technical White Paper. April 1996.
13. Dang, Winston: Reliable File Transfer in the Multicast Domain. 1993. ftp://ftp.hawaii.edu/paccom/imm-3.3/ Accessed Mar. 21, 2003.
14. Adamson, B.; and Macker, J.: The Multicast Dissemination Protocol (MDP). Naval Research Laboratory MDP Protocol Specification Version 1.6, 1999. http://manimac.itd.nrl.navy.mil/MDP/DraftMdpSpec-1.6.txt
15. STS-99 (97). http://science.ksc.nasa.gov/shuttle/missions/sts-99/mission-sts-99.html Accessed Mar. 21, 2003.
16. Mathis, Matthew, et al.: The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm. Computer Communication Review, vol. 27, no. 3, July 1997. http://www.acm.org/sigcomm/ccr/ Accessed Jan. 8, 2003.
17. Tierney, Brian L.: TCP Tuning Guide for Distributed Application on Wide Area Networks. http://www-didc.lbl.gov/TCP-tuning/ Accessed Jan. 8, 2003.
18. Xiphos Technologies. http://www.xiphos.ca/ Accessed Jan. 8, 2003.
19. Skipware. http://www.skipware.com/ Accessed Jan. 8, 2003.
20. Space Communications Protocol Specification (SCPS)—Transport Protocol (SCPS-TP). CCSDS 714.0–B–1 Blue Book, May 1999.
21. Mo, J., et al.: Analysis and Comparison of TCP Reno and Vegas. Proceedings of IEEE Infocom '99—The Conference on Computer Communications, vols. 1–3, 1999, pp. 1556–1563.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	May 2005	Technical Memorandum

4. TITLE AND SUBTITLE

SCPS–TP, TCP, and Rate-Based Protocol Evaluation

5. FUNDING NUMBERS

WBS–22–322–20–01

6. AUTHOR(S)

Diepchi T. Tran, Frances J. Lawas-Grodek, Robert P. Dimond, and
William D. Ivancic

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
John H. Glenn Research Center at Lewis Field
Cleveland, Ohio 44135–3191

8. PERFORMING ORGANIZATION REPORT NUMBER

E–13673-1

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Washington, DC 20546–0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

NASA TM—2003-212001-REV1

11. SUPPLEMENTARY NOTES

Diepchi T. Tran, Frances J. Lawas-Grodek, and William D. Ivancic, NASA Glenn Research Center; Robert P. Dimond, RS Information Systems, Inc., Brook Park, Ohio 44142. Responsible person, Diepchi T. Tran, organization code RCN, 216–433–8861.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified - Unlimited
Subject Category: 17

Available electronically at http://gltrs.grc.nasa.gov

This publication is available from the NASA Center for AeroSpace Information, 301–621–0390.

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 words)***

Tests were performed at Glenn Research Center to compare the performance of the Space Communications Protocol Standard Transport Protocol (SCPS–TP, otherwise known as "TCP Tranquility") relative to other variants of TCP and to determine the implementation maturity level of these protocols, particularly for higher speeds. The testing was performed over reasonably high data rates of up to 100 Mbps with delays that are characteristic of near-planetary environments. The tests were run for a fixed packet size, but for variously errored environments. This report documents the testing performed to date.

14. SUBJECT TERMS

Space communication; Protocol (computers); Communication networks; Telecommunication

15. NUMBER OF PAGES

52

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	